Spring 2019

Cloud Migration of RPAT Tool

Tanmay Gore
tgore03@iastate.edu

CREATIVE COMPONENT REPORT

CLOUD MIGRATION OF RIGID PAVEMENT ANALYSIS TOOL (RPAT)

TANMAY GORE

DEPT. OF COMPUTER SCIENCE

IOWA STATE UNIVERSITY

TABLE OF CONTENT

ABSTRACT

Machine learning has recently gained popularity in many engineering, science, medical and other domains outside computer science. Therefore, many researchers, scientist, students and developers are developing machine learning based applications for various purposes. However, due to a large number of technologies and application deployment platforms, these professionals spend more time learning technologies than on developing and optimizing the core logic for their applications. This paper describes the design and implementation of a new cloud-based deployment platform suitable to deploy machine learning based applications across multiple platforms. This platform focuses heavily on security, privacy, and ease of deployment for developers. It enables developers to define the application logic in any programming language without worrying about performance, scalability and other deployment-related issues. This paper further demonstrates how to develop these applications by migrating the Rigid Pavement Analysis Tool developed by Prof. Halil Ceylan and his team.

Keywords: Machine Learning, System Design, AWS Cloud, Security, Privacy, Platform-Independence, Application Deployment, Android Application.

1. INTRODUCTION

Machine Learning today is finding its application in a variety of areas outside Computer Science like Engineering, Construction, Medicines and many more. Its ability to predict values and identify hidden patterns in data proves valuable in these domains to simplify certain operations that are computationally complex. Therefore, people from these domains are increasingly using machine learning to solve their problems.

However, deploying machine learning based applications is very difficult due to high computational demands. Applications today need to be cross-platform, scalable, efficient and easy to maintain. These limitations and requirements cause the researchers and industry professionals to spend more time learning different technologies than developing these applications. They need to learn multiple languages, server management, database systems, networking and many other complex technical concepts which makes the development difficult.

This project, therefore, develops a cloud-based application deployment framework suited for professionals with limited advanced computer skills. It enables one to deploy cross-platform, highly scalable, secure apps in a very short amount of time. The platform considers different client requirements like privacy, security, cross-platform support, scalability, flexibility, low-cost and many more. These professionals only need to know one program language, one UI technology, and one database system.

Rigid Pavement Analysis Tool (RPAT) is a tool created by Prof. Halil Ceylan and his team that uses machine learning to predict the force exerted by different airplanes while landing on the runway. This is a Windows-based tool developed

using Visual Studio and C#. Being a Windows-based tool, it is very difficult to update, distribute and control its access.

This architecture was therefore designed to help migrate this and other such tools to cloud for rapid prototyping and deployment. This report further explains how the tool is migrated to the cloud architecture defined above and studies its performance.

2. DESIGN GOALS

Here we understand the different requirements that need to be considered when designing a single deployment platform for different applications

2.1. Cross-Platform Support

The application deployed over cloud should be able to interact with different end devices or platforms like Websites, Web Apps, Mobile Apps, Apps on different Operating Systems or even sensors.

2.2. Privacy and Total Control

Most research and corporate projects require the code and data to be confidential. Therefore, they need to be in complete control of the application and the data used by the application.

2.3. Security

End-to-End security is a need for all applications irrespective of their privacy requirements. End-to-End security means all communication between end-devices and server components should not be readable to an unauthorized user.

2.4. Support Multiple Languages

Different developers should be able to use different programming languages to define the server logic.

2.5. Scalability

The applications deployed on this platform should be able to achieve the same or similar performance irrespective of the task by scaling the compute resources. It should also handle multiple user requests made simultaneously.

2.6. Flexibility

The platform should be able to handle most of the client requirements like different databases, different data types and different programming languages.

2.7. Customizability

The platform should be easy to understand so that developers could modify it based on their requirements. These customizations should, however, try not to compromise privacy, security and cross-platform support.

2.8. Availability

The application deployed should be available to users with minimum download times. The architecture should be able to handle real-time failures caused by the AWS or by the application.

3. TERMINOLOGIES

3.1. Cloud Computing

"Cloud computing is the delivery of computing services – servers, storage, databases, networking, software, analytics, intelligence and many more – over the internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale" [1].

It follows the pay-per-use model, where the customer pays only for the services and resource they use.

3.2. User Interface (UI)

User Interface or Graphical User Interface abbreviated as UI or GUI is a tool or software that facilitates information exchange between user and computer. It takes input from users for processing and provides processed results back to users. Websites, Web Apps, Mobile Apps and software on other operating systems are some of the tools which contain a UI. Software and Tools with UI are often considered as part of End Devices.

3.3. REST API

Application Program Interface abbreviated as API is a documented method of interacting with some service provided by some developer.

Representational State Transfer abbreviated as REST is an architectural style used to distribute data of different forms in a standard format across different devices. It is widely used to transfer data over the internet using HTTP/HTTPS protocols.

3.4. Amazon Web Services (AWS)

"Amazon Web Services (AWS) is a secure cloud services platform, offering compute power, database storage, content delivery, and other functionality to help businesses scale and grow" [2].

3.5. Lambda Function

"AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume - there is no charge when your code is not running" [3].

It supports multiple languages, thus allowing the user to define the block of code in different languages.

3.6. API Gateway

"Amazon API Gateway is an AWS service that enables you to create, publish, maintain, monitor, and secure your own REST and WebSocket APIs at any scale. You can create robust, secure, and scalable APIs that access AWS or other web services, as well as data stored in the AWS Cloud" [4].

3.7. Virtual Private Cloud (VPC)

"Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS" [5].

3.8. Relational Database System (RDS)

"Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks" [6].

3.9. Simple Storage Service (S3)

Simple Storage Service Abbreviated as S3 is a document storage system used to store huge files that are uniquely identifiable.

"Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers" [7].

3.10. AWS Regions

AWS duplicates its services across 17 regions spread across the world. This helps AWS to provide the services with reduced latency.

3.11. AWS Availability Zones

Each AWS Region has two or more availability zones which are two or more isolated data centers. Availability zones are often replicated to ensure AWS services are available even in case of disasters or failures.

3.12. Security Groups

Security Groups define the security policy for AWS services. Each service has one security group. It acts as a firewall at service level where the user can define the type of services that can communicate with services with a security group and vice versa.

3.13. Route Tables

Route Tables are used to define the network topology within VPC. They tell AWS how to flow traffic from one service to another within VPC.

3.14. Access Control List (ACL)

Access Control Lists are firewalls that act on the traffic and port level. They are often applied at subnet level where they can control the flow inside/outside subnets. For example, ACL can allow traffic of HTTP/HTTPS type through port 8080.

3.15. Permissions

Permissions are given to the compute units in AWS. Developer grants these permissions to allow the program to act on behalf of a user. Permission can be given to Lambda Functions or EC2 instances to access or manipulate services within AWS for the user.

3.16. PCC Slab

"A composite stone-like material formed by mixing an aggregate (such as stones of irregular shape or crushed rock) with cement (which acts as the binding material) and water, then allowing the mixture to dry and harden" [8]. It forms the top layer of a rigid pavement system.

3.17. Subbase

"In highway engineering, subbase is the layer of aggregate material laid on the subgrade, on which the base course layer is located. It may be omitted when there will be only foot traffic on the pavement, but it is necessary for surfaces used by vehicles" [9]. There could be multiple subbase layers between the PCC Slab and Subgrade.

3.18. Subgrade

"Subgrades are commonly compacted before the construction of a road, pavement or railway track, and are sometimes stabilized by the addition of asphalt, lime, Portland cement or other modifiers. The subgrade is the foundation of the pavement structure, on which the subbase is laid" [10].

3.19. E Modulus

It stands for elastic modulus which defines the elasticity of all the layers in the pavement system. It can be calculated by obtaining the slope of the linear portion of the stress-strain curve.

3.20. Poisson Ratio

It is defined as the ratio of the longitudinal and transverse strain. It measures the expansion of a material in a direction perpendicular to the direction of applied force.

3.21. Thickness

Thickness defines the depth of each layer in the pavement system.

3.22. Gear Position – X & Y Offset

Gear positions in x and y offsets show relative locations of the center of gear loads compared to slab center. As x offset increases, the center of the gear moves toward the edge of the slab in x-direction. Similarly, as y offset increases, the center of the gear moves toward the edge of the slab in y-direction.

3.23. Gear Position - Loading Angle

Loading angle shows angle value between gear direction and when the gear is parallel to y-axis (gear parallel to the y-axis is assumed as having 0° loading angle). Gear angles vary from 0° (gear parallel to the y-axis) to 90° (gear parallel to the x-axis)

3.24. Joint Stiffness

Adjacent slabs are connected to each other with springs to transfer load from one slab to another. Joint Stiffness measures the transfer of load. As joint stiffness increases, there is a higher level of load transfer between slabs.

3.25. Temperature Gradient

It defines the rate of change of temperature from the top to the bottom of the pavement system.

3.26. Thermal Coefficient

Due to changes in temperature, the pavement system undergoes expansion or contraction which subsequently results in strain. The Coefficient of Thermal Expansion or Thermal Coefficient measures the strain produced due to temperature changes.

4. DESIGN

For the design & implementation of the system, AWS Documentation was extensively referred [11].

4.1. HIGH-LEVEL COMPONENT INTERACTION

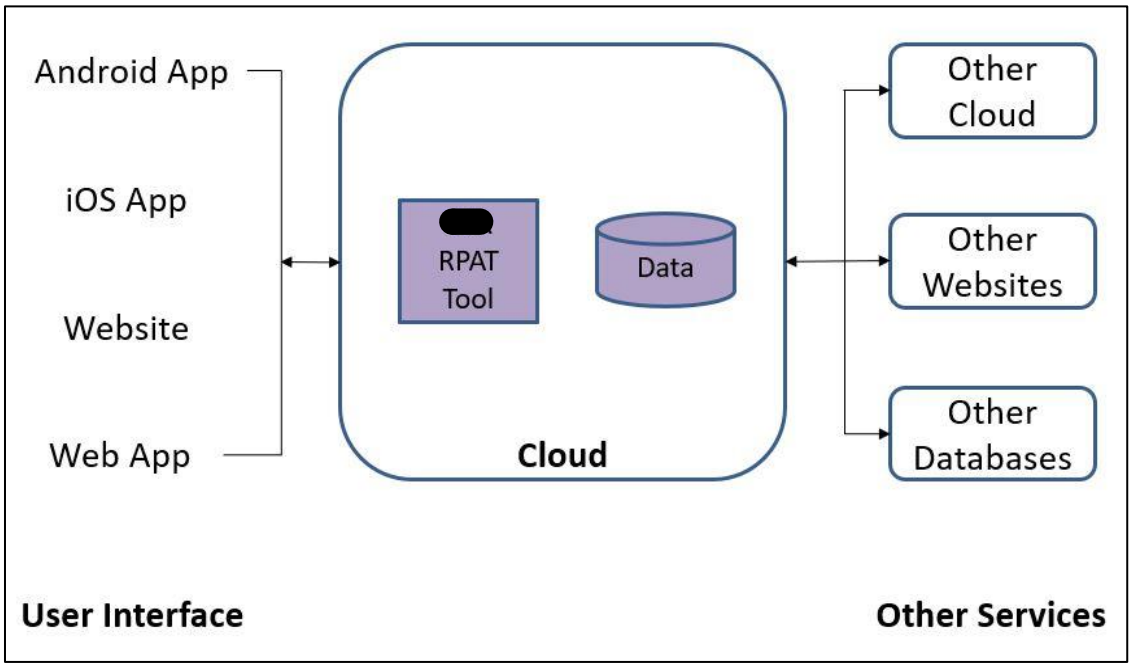


Figure 1: High-Level Component Interaction

The entire system is divided into 3 logical parts.

4.1.1. User Interface

This serves as a dumb end-device who is responsible to obtain inputs from users and feed the inputs to Cloud for processing. The processed results are then received from Cloud which it displays to the User.

4.1.2. Cloud

The cloud forms the brains for the application. It is responsible for all data processing tasks like preprocessing of data, executing

operations on data and post-processing of results. The Cloud can also communicate with other clouds, websites or databases.

4.1.3. Other Services

Other services include services or resources provided by the third party that is required by the application. Other services could be a source of data like databases, other end devices like websites or other application already deployed on other cloud services.

This architecture thus uses a Client-Server architecture where the client is the UI and the Server is the Cloud. It helps developers to customize their applications by choosing any UI technology and any Cloud technology they are familiar with.

4.2. LOW-LEVEL COMPONENT INTERACTION

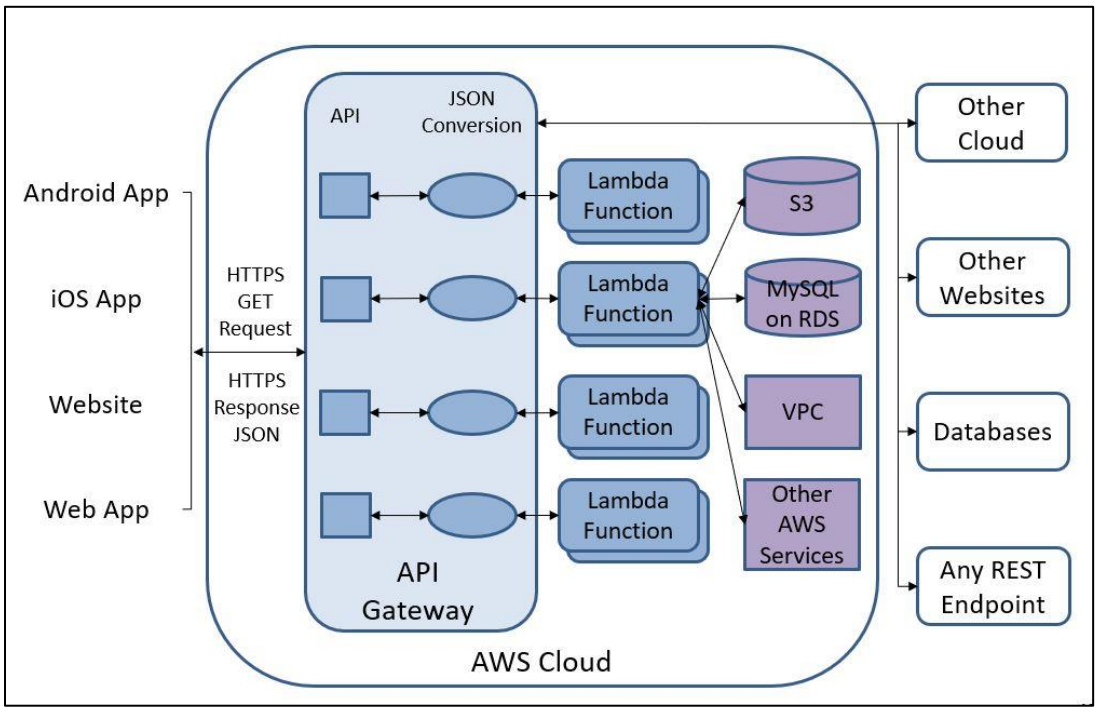


Figure 2: Low-Level Component Interaction

The Cloud is further divided into 3 logical parts

4.2.1. API Gateway

API Gateway is the internet facing unit which obtains the data from various UI devices. API Gateway receives an HTTP Request sent over the internet using HTTP/HTTPS.

The API Gateway is responsible for the following tasks:

- **Translating HTTP requests to Lambda Function Input**

  To understand the input and extract relevant information, API Gateway has APIs defined. These APIs helps the gateway to understand what request to expect from UI Device and how to extract the relevant information from the request.

  As shown in *figure 2*, the API Gateway is receiving an HTTPS GET Request, therefore, there must be an API that can parse this request. The Request looks like *https://www.example.com/example_folder?input1=input_data1&input2=input_data2*. Therefore, an API exists which accepts an HTTPS GET Request sent to *https://www.example.com/example_folder* location. The API further has logic defined to extract input_data1 and input_data2 from the input1 and input2 variables passed.

  The request can vary based on the type of request like GET, POST, PUT and DELETE and number and types of inputs passed. Therefore, an API should exist that can handle the request type and all the inputs passed in it. This structure of a request is called as REST which helps to communicate with any UI Devices in a standard format.

  The APIs later convert the extracted input data into JSON and sends it to the Lambda Function. API Gateway provides a GUI to develop these APIs and link them to the appropriate lambda

function. Thus, the APIs know exactly what inputs it receives and what output it should to whom.

- **Translating Lambda Function result in HTTP Response**

  Once the lambda function completes processing the inputs, it might return a result in JSON format. The API then converts the JSON back into HTTP/HTTPS Response which is then sent to the UI device via HTTP/HTTPS protocol. The Response also contains status id which helps the UI device know if the request was correctly processed else what error occurred. Response status 200 is sent if the request is correctly parsed else id of the error is sent.

- **Authentication and Validation of the Request**

  For added security API Gateway is a response to authenticate the sender of HTTP Request. It can authenticate a request using AWS Identity and Access Management Profile (IAM) or using API Key which is a unique key for an API. The API gateway further validates the request by checking the variable data for injection attacks.

- **Rerouting requests**

  API Gateway is also capable to forward certain HTTP Requests to other clouds, websites or any other REST Endpoint capable of handling the request. In this case, an API exists which contains logic to reroute the request to third party handlers. This enables seamless integration of pre-existing systems into the application. Thus, allowing us to reuse the old system without any modifications.

4.2.2. LAMBDA FUNCTIONS

Lambda functions are the computing units where the logic for processing the inputs from APIs is located. They are the most important unit for rapid deployment as they provide the following benefits:

- **Multi-Language Support**

  Lambda functions can execute code written in C#, Go, Java, Node.js, Python, and Ruby providing developers the flexibility to code in their language of choice.

- **Scalability**

  Lambda Functions are Serverless computing units, therefore, AWS can spawn multiple copies of lambda functions simultaneously to handle simultaneous HTTP Requests made to one API. This enables concurrent multi-user support which automatically scales depending upon the demand.

- **Modularization of Logic**

  Since all lambda functions take JSON input and return JSON output, they can communicate with each other allowing developers to modularize their logic and enable parallel processing for tasks.

- **Access AWS Services**

  Lambda Functions have the capability to access any AWS services like S3, RDS, and Lambda allowing developers to easily use any technology they require and integrate it within their application. Most of the services allow users to migrate their existing logic and data into AWS provided service and since these services are located on the cloud, they can meet the scalability requirements of the application.

4.2.3. AWS Services

AWS provides multiple services like Relational Databases, Non-Relational Databases, Publish-Subscribe based Communication, Security and many more which are easily accessible from within AWS. The access to these services is often controlled using permissions to prevent unauthorized access from within the cloud. These services could be directly exposed to the internet or could be hidden within the cloud for privacy. For example, keeping a database hidden with the cloud with access only using Lambda functions or EC2 instance can provide privacy and security of data by preventing unauthorized access.

4.3. HIGH-LEVEL CLOUD ARCHITECTURE

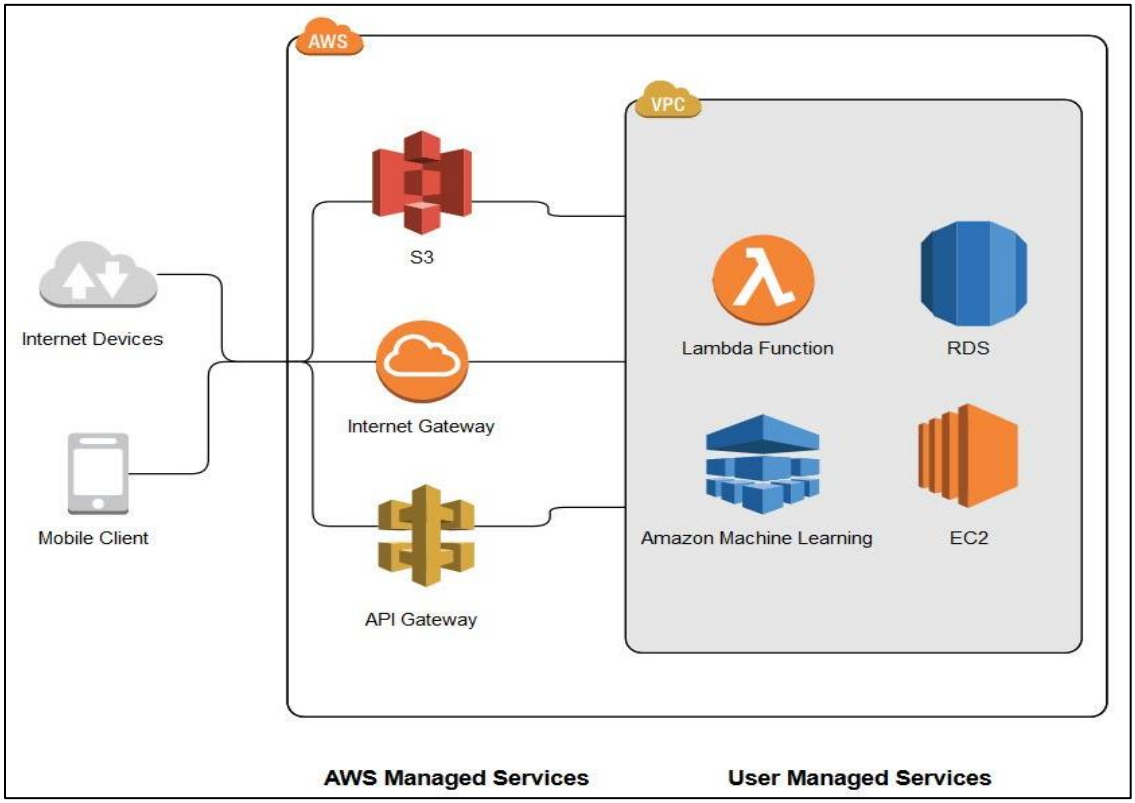


Figure 3: High-Level Cloud Architecture

The cloud architecture is defined to satisfy all the design goals of the project. The whole architecture is divided into 3 components as follows.

4.3.1. UI DEVICES

These are the devices/tools that enable users to interact with the application or the system. It consists of Apps and Websites that can communicate with AWS over the internet.

4.3.2. AWS MANAGED SERVICES

These are the services provided and managed by AWS for the user. They are placed outside the Virtual Private Cloud and can be accessed in a secure fashion using Endpoints. These services are outside VPC because they either need to be internet facing or AWS infrastructure limits their placement within VPC.

Internet Gateway is a service that enables other services within VPC to communicate outwards via the internet. S3 provides object storage capability which is often used for website deployment. Hence it is placed outside and communicates with services within VPC using VPC-Endpoint. API Gateway being the internet facing service is managed by AWS, hence, cannot be within a VPC.

4.3.3. USER MANAGED SERVICES

These are the services provided by AWS but are managed by the User/Developer. Therefore, they can be placed within the Virtual Private Cloud. These services receive extra protection from Internet-based attacks as they are stored within a private cloud architecture that enforces additional protection for traffic flowing in/out of VPC and within VPC. Each service is guarded by its own firewall within the VPC.

RDS, EFS (Elastic File System), EC2 (Elastic Cloud Compute) are the few services that users manage. Lambda Functions are special functions as they can be placed within and outside the VPC as they are partially managed by AWS.

4.4. DETAILED CLOUD ARCHITECTURE

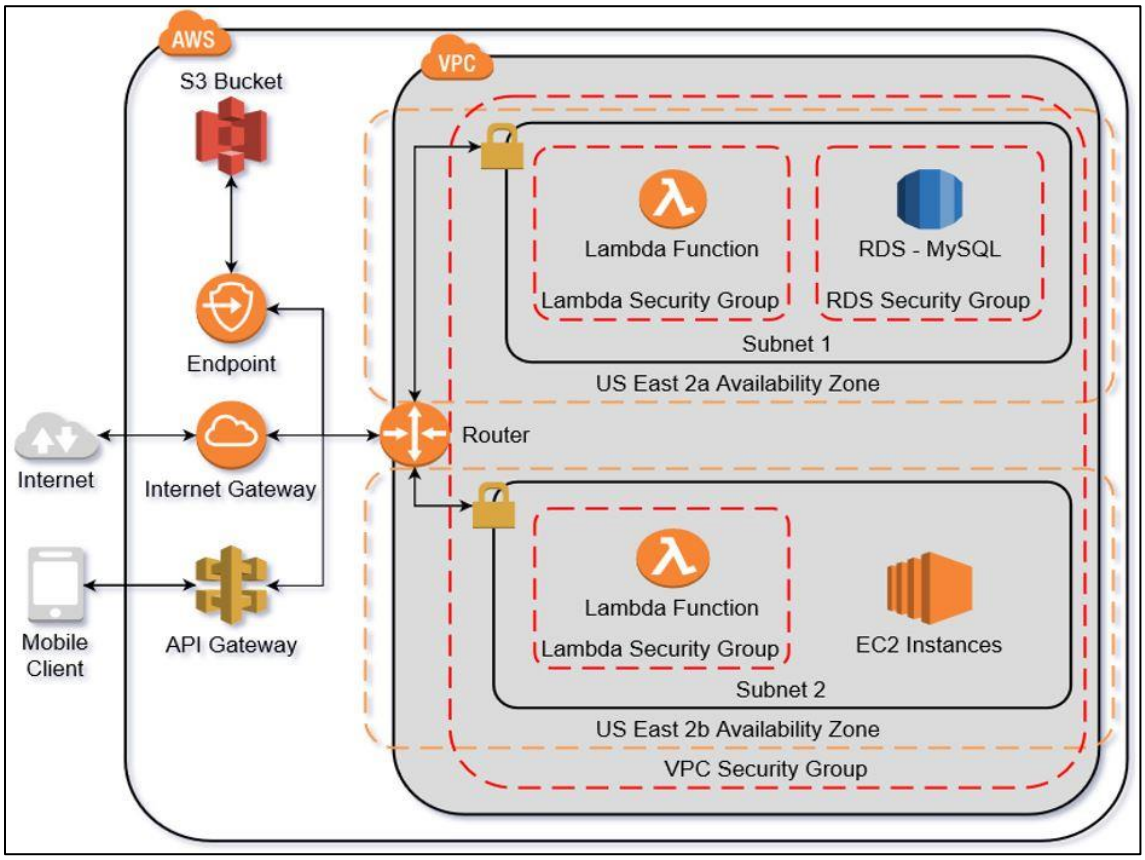


Figure 4: Detailed Cloud Architecture

Here we expand on the Architecture of AWS managed services and the User managed services. The gray area defines the architecture within VPC. Yellow dotted lines define the services within Availability Zones. Red dotted lines define the security group. Note VPC has its own security group. Therefore, services within VPC that do not use their own security group inherits the security group of the VPC.

4.4.1. AWS MANAGED SERVICES

4.4.1.1. Router

Each of the services managed by AWS and/or placed outside VPC has a public IP Address associated with it. The router acts as the entry point into the VPC. It uses the route table of the VPC to decide where to send the message.

If the message is marked for delivery to a public IP address, it then takes it out of the VPC to either send it to Internet Gateway, the Endpoint or the API Gateway. The destination depends on the IP address provided in the message.

If the message is marked for delivery within the VPC, then the message never leaves the VPC. The router can allow messages from one subnet to be delivered to a service in another subnet.

Once the message is correctly routed, it is then sent to the security group or authentication engine to validate the access.

4.4.1.2. S3 Bucket

Since S3 is placed outside VPC, AWS provides an endpoint for services in VPC to access S3 bucket in a secure fashion. Endpoint acts like a VPN tunneling the traffic to S3 in an encrypted format. Therefore, messages intended for S3 bucket are routed to the endpoint created for S3.

4.4.1.3. API Gateway

API Gateway has the APIs defined by the developer to accept the inputs from the internet and forward it to the lambda function. Since lambda is partially managed by AWS and is created on the occurrence of an event, AWS first creates a lambda function within or outside the VPC depending on its configuration. The router then forwards the message from API

Gateway to lambda within VPC. Since AWS creates the lambda, it shares the private/public IP address of lambda with the router for correct routing.

4.4.2. USER MANAGED SERVICES

The architecture within VPC is specially designed for design goals like security, flexibility, privacy, and availability. For the RPAT project, VPC only allows HTTP/HTTPS and MySQL traffic within VPC.

4.4.2.1. Subnets

The VPC is divided into 2 subnets each deployed in two different availability zones. Every service within a subnet is duplicated into another subnet to provide high availability of services and reduce the downtime of the application. Each subnet sends traffic to the router for further routing and they control the traffic it receives and sends using the Access Control List (ACL).

4.4.2.2. Lambda Functions

Since lambda functions are the core processing units of the application and need to be highly available, they need at least two subnets in different availability regions for their creation. AWS enforces this restriction to ensure the application does not fail in case of failures. This restriction only exists when they are created within VPC as lambda functions outside VPC could be created in any availability zones.

Lambda Function has its own security group and set of permissions as the security for lambda function depends upon the application created by the developer. The permissions of the lambda function should only allow access to services that

program in lambda function is using. Permissions can be as specific as certain methods/functions that the service provides.

4.4.2.3. RDS

Since RDS holds the database which contains the application data, it is placed within the VPC. RDS has the capabilities to replicate itself in two availability zones for high availability and loss of data. However, AWS charges extra for this feature, hence, was not used for the RPAT project.

RDS also has its own security group to customize the security rules to the service.

4.4.2.4. EC2 Instances

EC2 or Elastic Cloud Compute is a virtual computer created within AWS. It provides a back-channel access to services deployed within VPC for developers to access and manage the VPC services without creating APIs and lambda functions. It is heavily secured by enforcing strict authentication mechanism ensuring only authorized users can access it.

Since EC2 is a computer within AWS that the developer can control from outside AWS, he can use it to debug applications, network configurations and test security within VPC. Developers can create EC2 instances anywhere within VPC using AWS Management Console, therefore, AWS Management Console credentials become the most important security resource that should not be shared with anyone. The credentials should only be shared with the developer the owner trusts and are responsible to manage the AWS architecture for the application.

EC2 inherits the security group of the VPC and has its own permissions. EC2 is given permission to access any service within AWS as a developer need them to manage the cloud.

5. IMPLEMENTATION

This section talks about how to implement the proposed design using the Concrete Rigid Pavement Analysis Tool as the example

5.1. AWS ARCHITECTURE SETUP

Following is the process to recreate the architecture. The bullets do not truly follow the order as these configurations are dependent on one another. For example, you cannot create a route for internet gateway without first creating an internet gateway.

5.1.1. VPC Setup

Create a VPC from the AWS Management Console [11] with IPv4 CIDR Block as *xxx.xxx.0.0/16*. Tenancy as "dedicated" allocates dedicate hardware for services in your VPC that are not shared with anyone but for a cost. Default shares the resources.

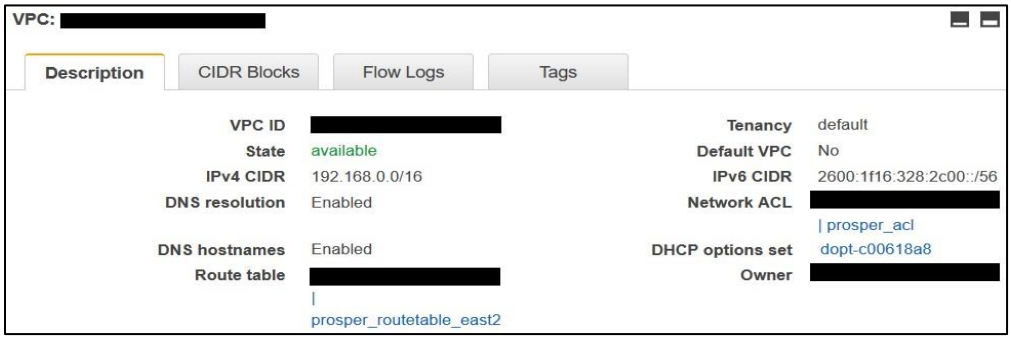


Figure 5: VPC Details

5.1.2. Subnet Setup

Next setup 2 Subnets. We created them in us-east-2a and us-east-2b availability zones. The IPv4 CIDR block should of format *xxx.xxx.0.0/24* and *xxx.xxx.1.0/24*. Here the "*xxx.xxx"* should be the

same as that of VPC. The CIDR block defines a range for the IP address that a service can have within that Subnet.

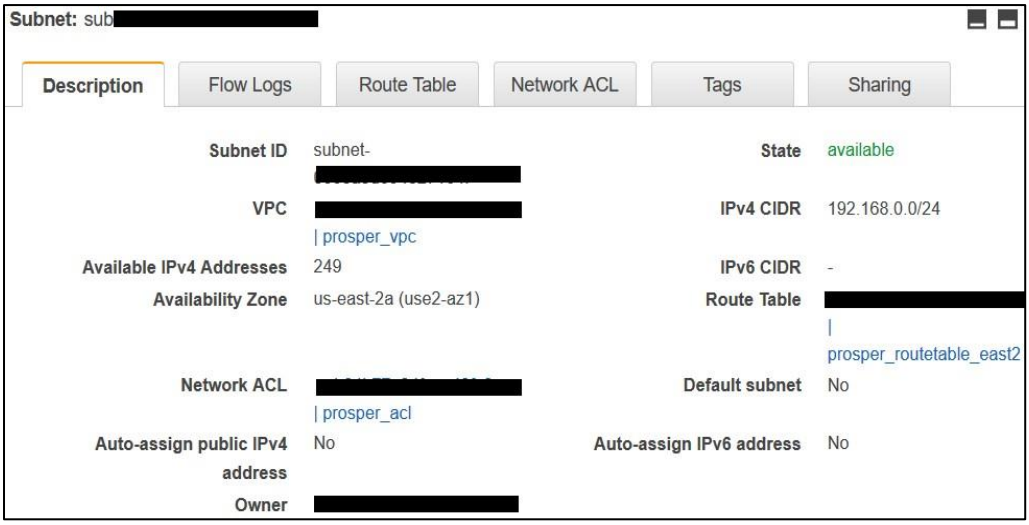


Figure 6: Subnet 1 Details

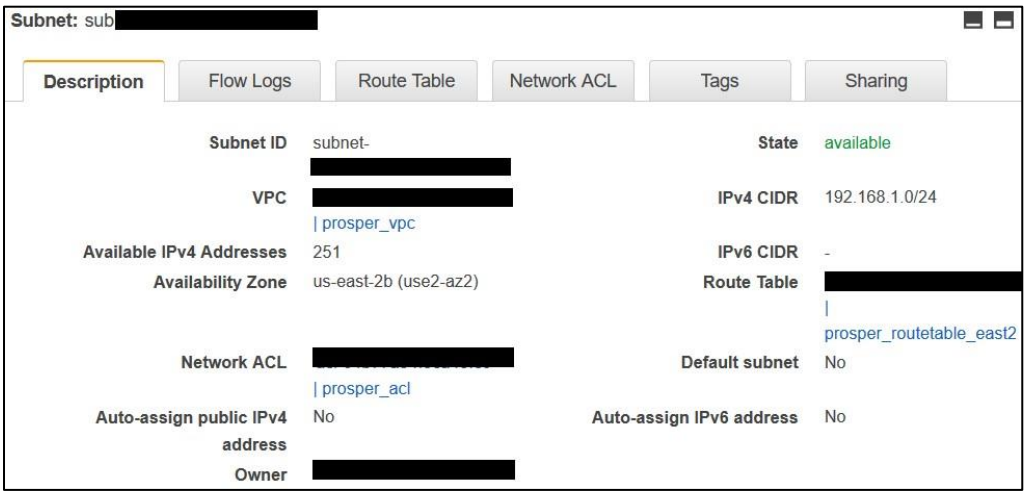


Figure 7: Subnet 2 Details

5.1.3. Route Table Setup

This defines how to route traffic within VPC. If your network configuration does not work, carefully check the routes defined.

First, create a Route table say *prosper_routetable*.

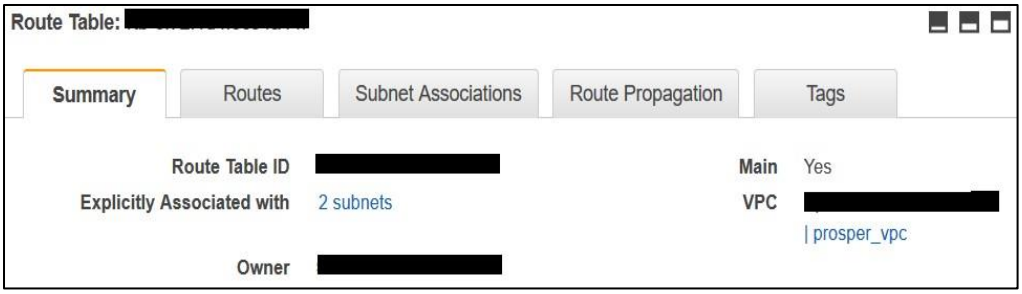


Figure 8: Route Table Details

Next, create routes within the route table. Since 192.168.0.0/16 is the CIDR block of VPC, any message with the destination IP 192.168.x.x will be present within VPC, therefore, will be routed locally. pl-xxx..x is the IP address of S3, therefore will be routed to Virtual Private Cloud Endpoint(VPCE) that we create during S3 creation. If the message is for any other destination, it will be forwarded to the Internet Gateway (IGW) to be sent to the internet. We create IGW later.

Destination	Target
192.168.0.0/16	local
2600:1f16:328:2c00::/56	local
pl-[redacted] (com.amazonaws.us-east-2.s3, [redacted])	vpce-[redacted]
0.0.0.0/0	igw-[redacted]

Figure 9: Route Table – Routes

Next, you need to associate this route table to the two subnets we created before.

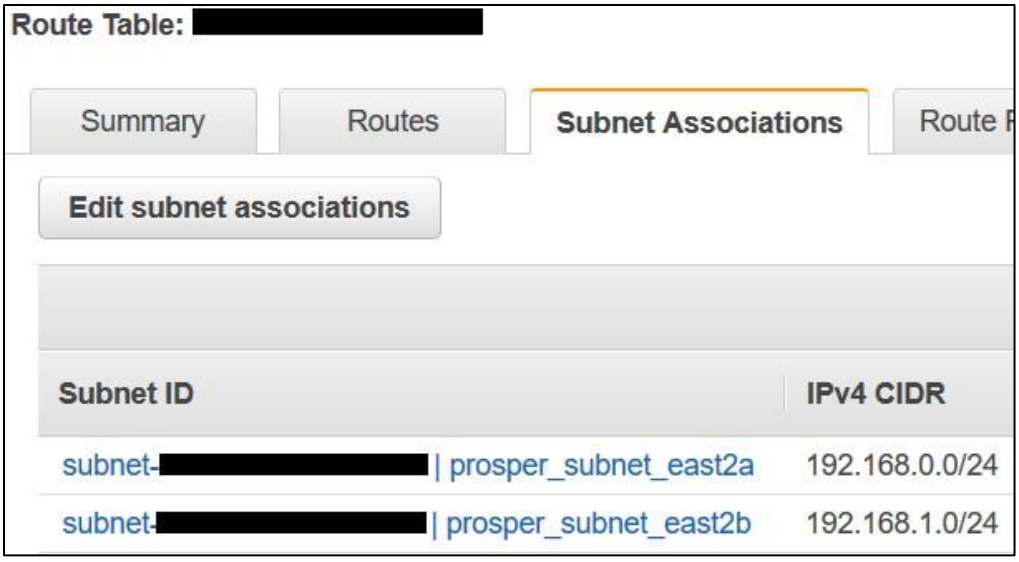


Figure 10: Route Table - Subnet Association

You use the Edit subnet association to add the subnets. If you created the route table correct VPC, these subnets will be visible.

5.1.4. Access Control List Setup

It defines the type of traffic allowed to communicate with resources within VPC. It acts as a firewall to prevent unwanted traffic. For the RPAT project, we create one ACL and link it to the VPC. This ACL also needs to be associated with the two subnets we have as the subnets are meant to be exactly the same for failure protection.

Figure 11: Access Control List – Details

We allow all protocols to communicate with the resources at all ports for flexibility. The security group of each service controls its access. This allows us to define the VPN with different services, hence provides flexibility but at the same time provides security via security groups.

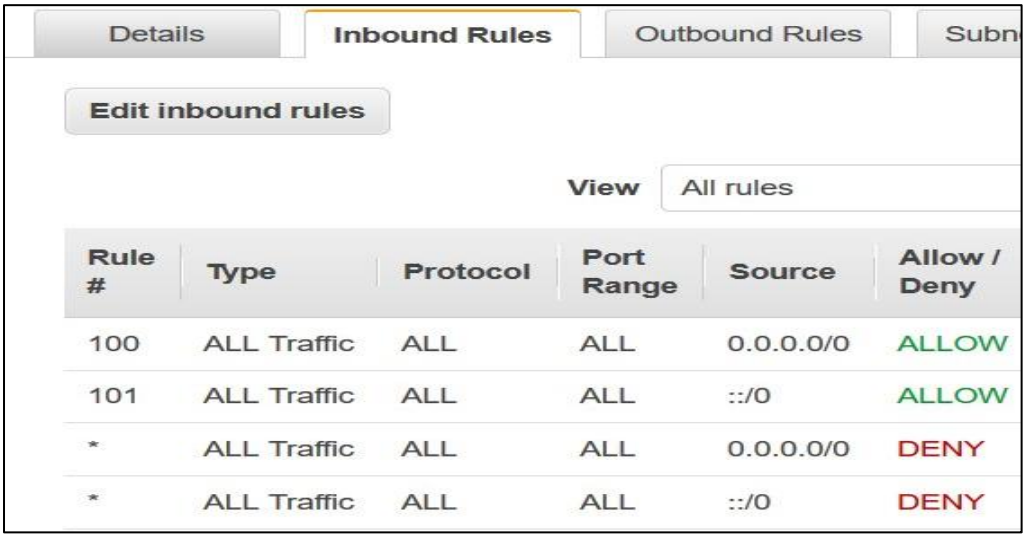


Figure 12: Access Control List - Inbound Rules

Similarly, we allow all services within VPC to communicate with any resources outside VPC via any protocol on any port. Similarly, we control the access for individual service within VPC via security groups.

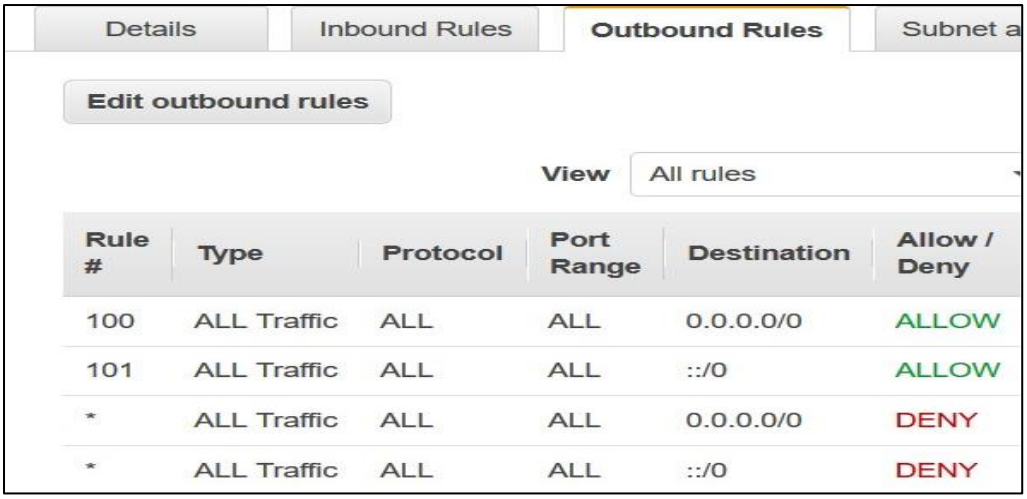


Figure 13: Access Control List - Outbound Rules

5.1.5. Security Group Setup

Security Groups define the type of protocol and the ports allowed to communicate with the service in both directions. VPC being a service has its own Security Group. If a service within VPC does not have its own security group, then it inherits the security group of the VPC. First, we create a security group for the VPC. When a new VPC is created, a security group is automatically created for it.

For the RPAT project, we create a security group called *security_group_prosper*.

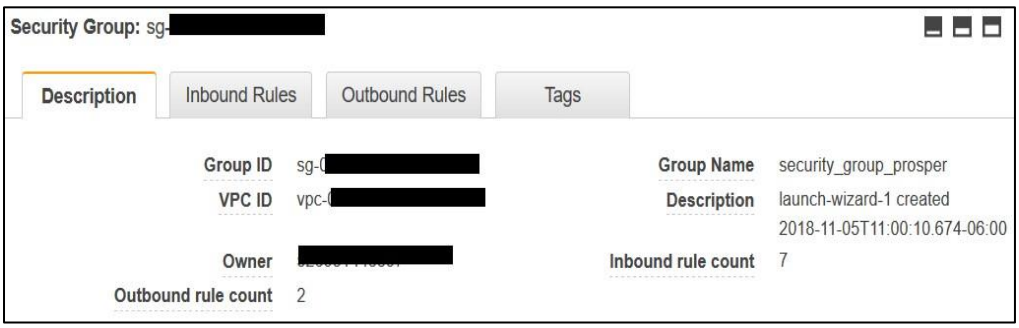


Figure 14: VPC Security Group – Details

Next, we define the Inbound rules for the security group.

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
All TCP	TCP	0 - 65535	192.168.0.0/16
SSH	TCP	22	[redacted]2
MYSQL/Aurora	TCP	3306	192.168.0.0/16
HTTPS	TCP	443	0.0.0.0/0
HTTPS	TCP	443	::/0

Figure 15: VPC Security Group - Inbound Rules

The VPC allows HTTP and HTTPS (Hyper Text Transfer Protocol Secure) traffic from anywhere on the internet. This helps to send/receive data via REST. The two HTTP rules define that traffic can use both IPv4 or IPv6 for communication. The VPC also allows all TCP (Transmission Control Protocol) type of traffic within the VPC. This facilitates communication between subnets and services within VPC. The VPC also allows traffic from MYSQL services defined within VPC. This allows lambda functions deployed within VPC to read/write data in Database. The MYSQL database will be created in later steps as it is used by the RPAT Project. Lastly, we allow SSH (Secure Shell) type of traffic within the VPC. This enables developers to create an EC2 instance within VPC and access it over the internet.

For other services, appropriate rules need to be set to allow communication. The outbound rules allow every type of communication. This is done for simplicity; however, appropriate rules could be set to prevent certain types of traffic to flow outside the VPC. This could provide an added level of security if an unauthorized user tries to copy content outside the VPC.

Type	Protocol	Port Range	Destination
All traffic	All	All	0.0.0.0/0
All traffic	All	All	::/0

Figure 16: VPC Security Group - Outbound Rules

5.1.6. RDS Setup

RPAT project uses RDS to store all the airplane and model information. It is used to only store the Metadata for the application while the actual data is stored in the S3 bucket. It helps to reduce the size of the database and hence helps for easy replication across

multiple locations and reduces the cost applied by AWS. First, we create an RDS database using MySQL engine, define the name as prosper_faa and create it in our VPC. We use the same security group as VPC.

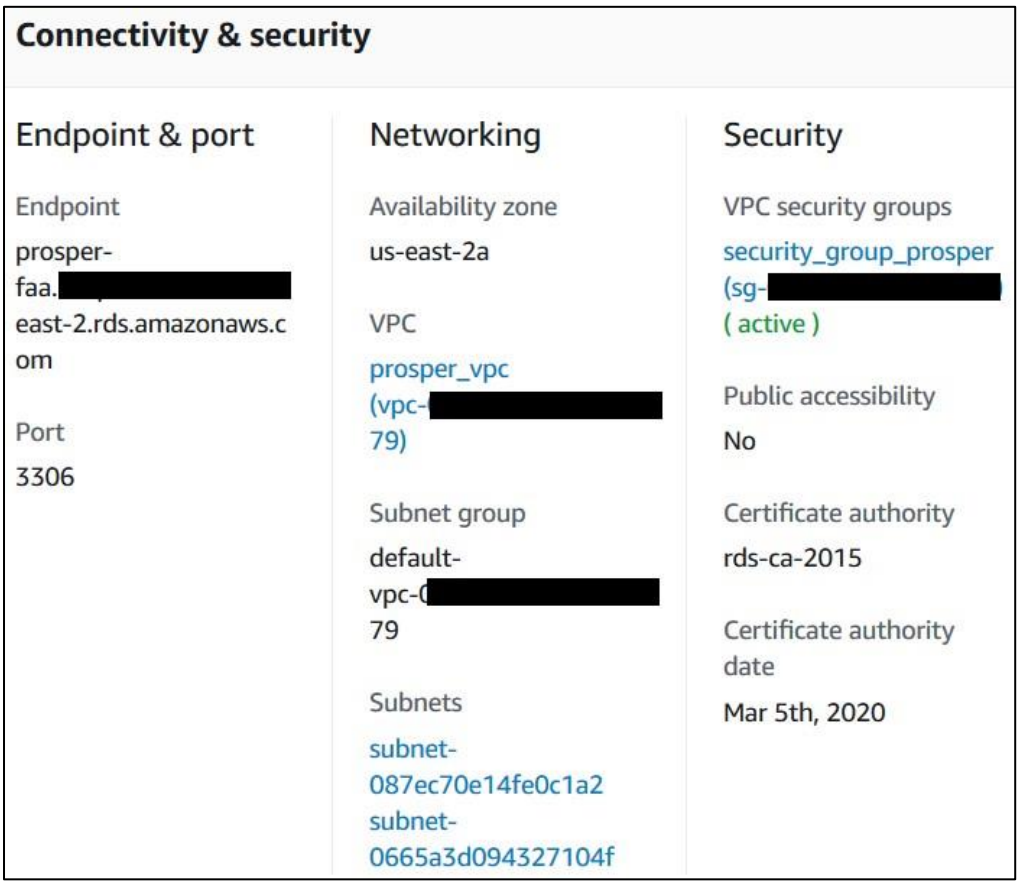


Figure 17: RDS – Details

Since we use the security group of the VPC, RDS can only be accessed by services within VPC making it very secure. This happens because in *Figure 15* we only allow MySQL type communication from within VPC. This enforces following security policy where 192.168.0.0/16 is the CIDR block of VPC.

Security group	Type	Rule
security_group_prosper ([redacted])	CIDR/IP - Inbound	192.168.0.0/16
security_group_prosper ([redacted])	CIDR/IP - Inbound	192.168.0.0/16
security_group_prosper ([redacted])	CIDR/IP - Outbound	0.0.0.0/0

Figure 18: RDS - Security Policy

Within RDS, the RPAT project has one database named “faa_ann”. This database contains 2 tables namely “models” and “plane_info” to store model information and airplane information respectively.

- Models Table

  The schema for the model table is

Field	Type	Null	Key	Default	Extra
model_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
model_file	varchar(100)	NO		NULL	
version_id	varchar(32)	NO		NULL	
model_type	varchar(10)	NO		NULL	
timestamp	timestamp	NO		CURRENT_TIMESTAMP	
train_acc	float(5,2)	YES		NULL	
test_acc	float(5,2)	YES		NULL	

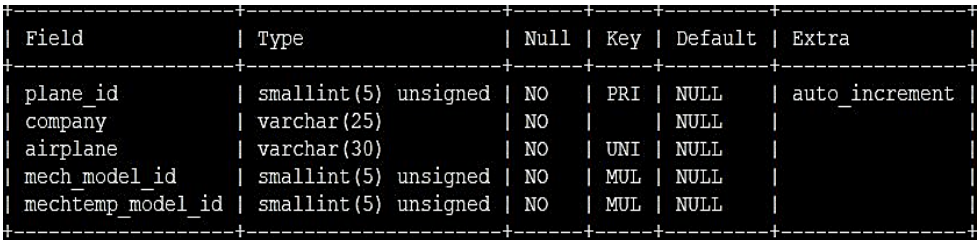
Figure 19: RDS - Model Table Schema

  - Model ID: Contains a unique ID to identify the model. This field is a primary key, therefore, cannot have null values and is auto incrementing.
  - Model File: Stores the S3 object ID of the model file stored in S3 Bucket.
  - Version ID: Stores the version number of the model file stored in S3. This allows us to update the model file without changing the model name.
  - Model Type: Defines if the model of type “mech + temp” or just “mech”.

- Timestamp: Contains information of when the model information was stored in the table. It is automatically entered.
- Training Accuracy: Provides training accuracy of the model which could be used in the future. This field is optional and therefore could be “Null”.
- Testing Accuracy: Provides accuracy of the model on test set which could be used in the future. This field is optional and therefore could be “Null”.

- Plane Info Table

  The schema for the plane_info table is



Field	Type	Null	Key	Default	Extra
plane_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
company	varchar(25)	NO		NULL	
airplane	varchar(30)	NO	UNI	NULL	
mech_model_id	smallint(5) unsigned	NO	MUL	NULL	
mechtemp_model_id	smallint(5) unsigned	NO	MUL	NULL	

Figure 20: RDS - Plane Info Table Schema

- Plane ID: Defines a unique identifier for the plane.
- Company: Stores the name of the company who manufactures the plane model.
- Airplane: Stores the airplane model name.
- Mech Model ID: Defines the Model ID from Models table of type “mech”.
- Mechtemp Model ID: Defines the Model ID from Models table of type “mech + temp”.

5.1.7. S3 Setup

In RPAT we use S3 for storing all the model files which are very large. Since S3 provides a unique ID for every file stored in it, we can use it along with RDS to easily retrieve the correct file. This allows us to store metadata of the files in the RDS while still maintaining pointers to the files in S3.

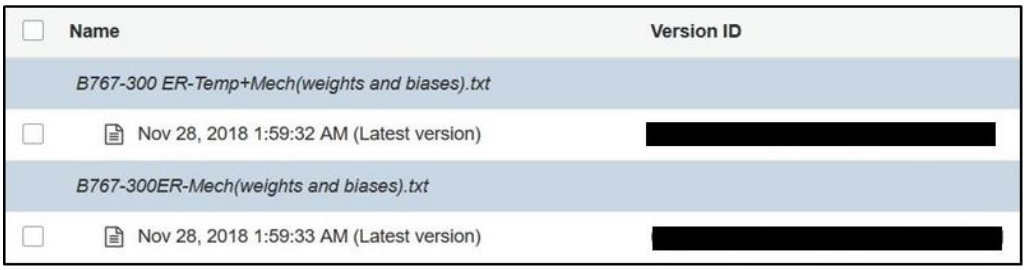


Figure 21: S3 - Bucket Content

Since the S3 bucket cannot be created in the VPC, as it is managed by AWS, we created a VPC Endpoint which allows us to access the S3 bucket like a VPN.

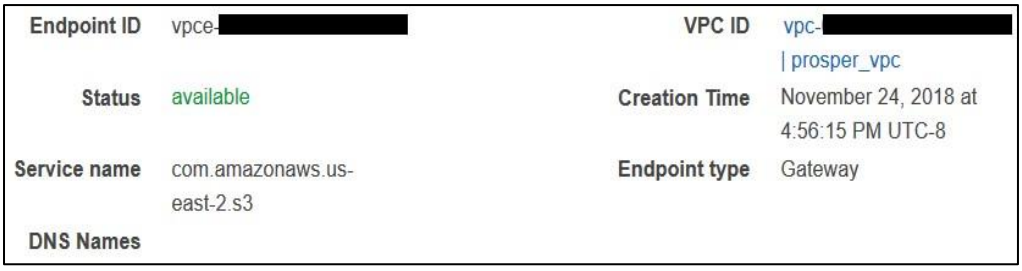


Figure 22: S3 - VPC Endpoint Details

5.1.8. EC2 Setup

Elastic Compute Cloud (EC2) is used by developers to debug and test their application and new service creation within VPC. It creates a virtual computer within VPC which the developer control controls remotely via SSH.

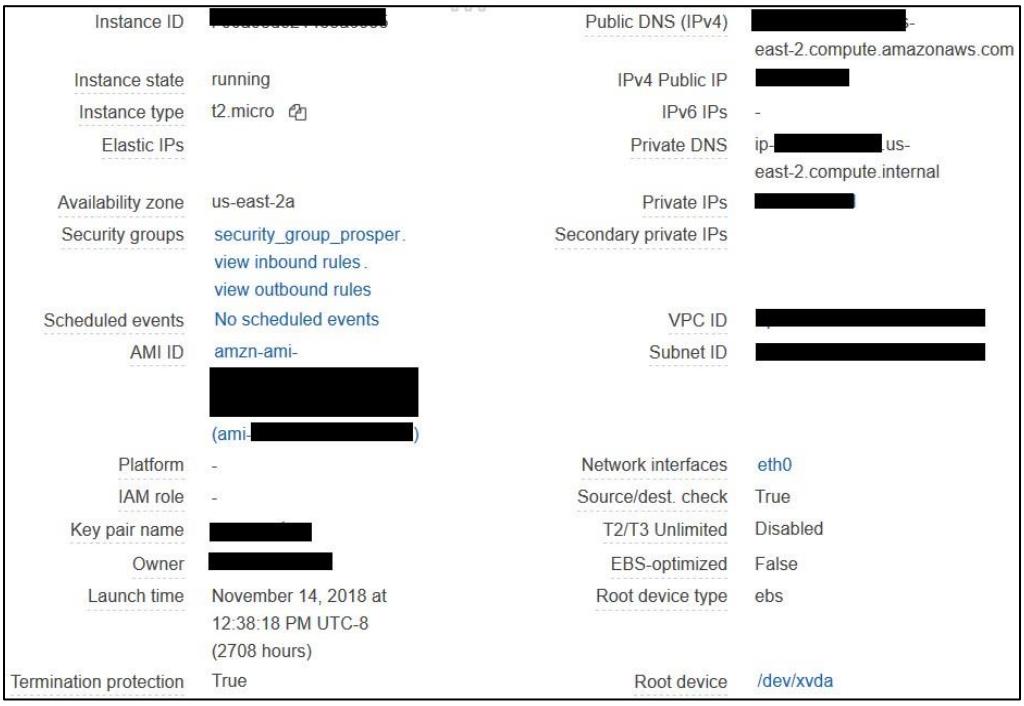


Figure 23: EC2 – Details

This EC2 instance inherits the security group of the VPC as it needs to manage all resources within the VPC. To connect to this virtual PC over SSH and control it, first, you need an SSH Tool like PuTTY and the Private Key for the instance. PuTTY takes the private and connects to the Public DNS. It later uses the private key for validation after which a command line window is available which is connected to the Virtual Computer.

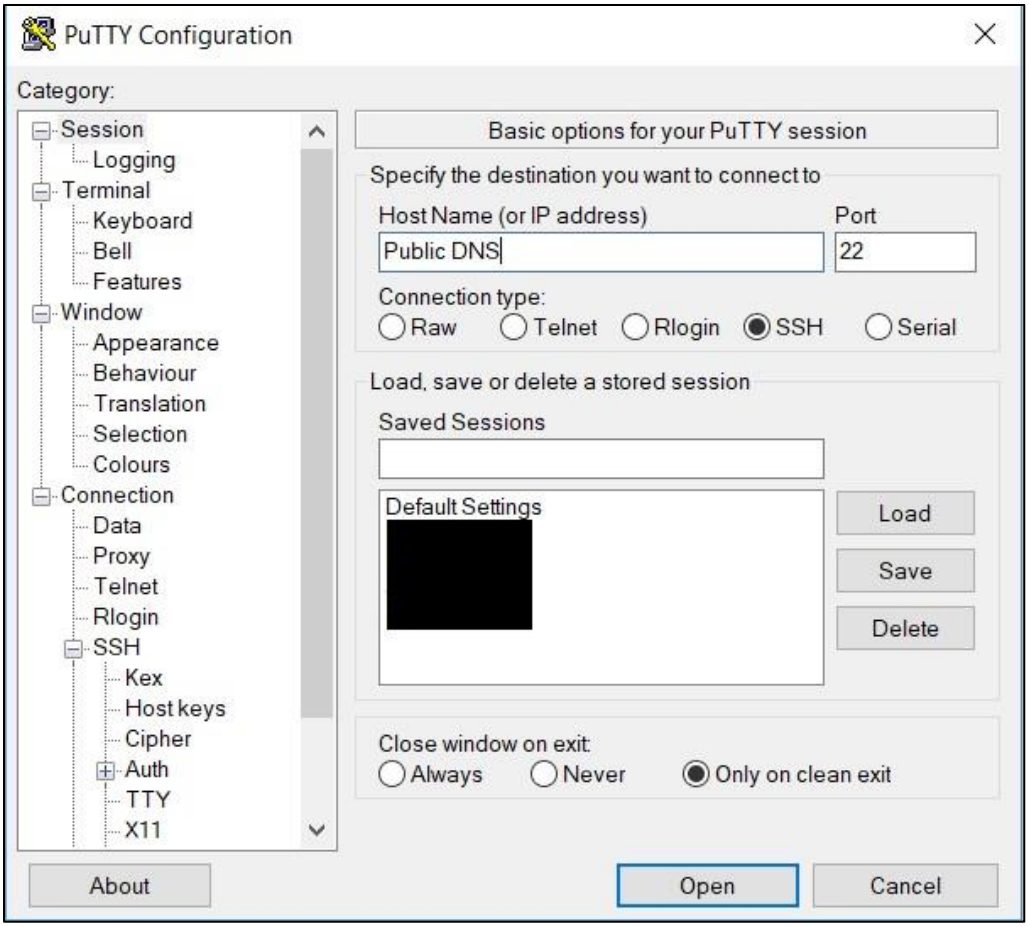


Figure 24: EC2 - PuTTY Configuration

Once the connection is validated and successful, we get the following command line.

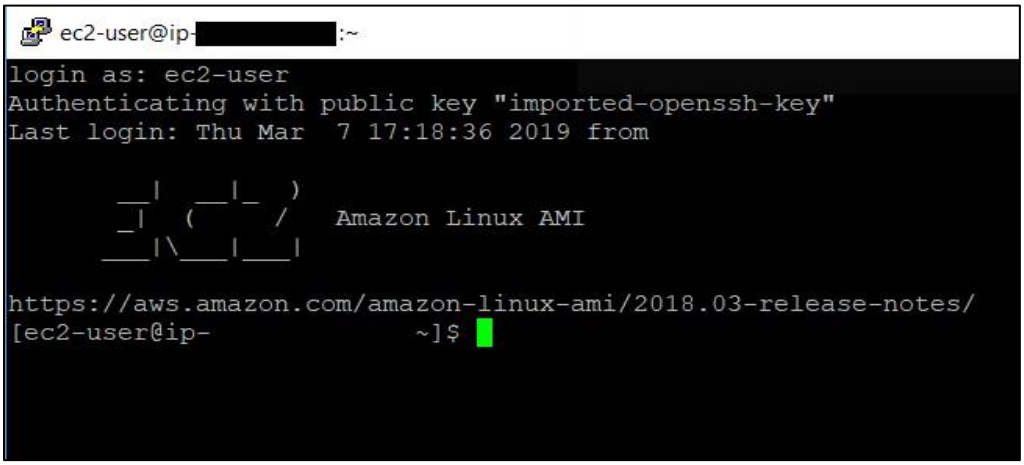


Figure 25: EC2 Connection

5.2. API AND LAMBDA SETUP

To enable UI Devices to communicate with AWS Services in a secure and standardized fashion we export an API using API Gateway. This API accepts requests in REST from the internet and sends it in JSON format to the lambda function. AWS provides a GUI to deploy new APIs and Lambda Functions.

5.2.1. API Creation

API Gateway provides a Graphical User Interface (GUI) to create APIs where you define the format of inputs to receive and the format of the outputs to send. First, we create a new API of type REST with Regional Endpoint Type.

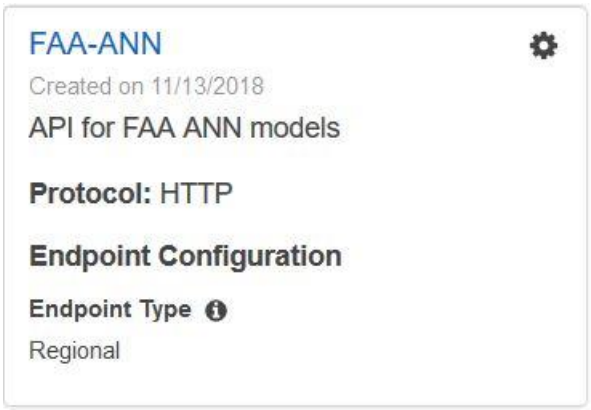


Figure 26: API Gateway - API Creation

Next, we need to create a resource within the API where our HTTP Request will be sent. For a resource with a name *resource_name*, the HTTP request will be sent to *www.xyx.com/resource_name*. Next, we need to define a method for receiving the request. This method defines the Type of HTTP Request to receive. The Method could be of type GET, POST, PUT, DELETE and other. We create a method of type GET.

Figure 27: API Method Creation

Next, we need to define the inputs this method passes and how to process them. For this AWS provides a GUI tool as follows.

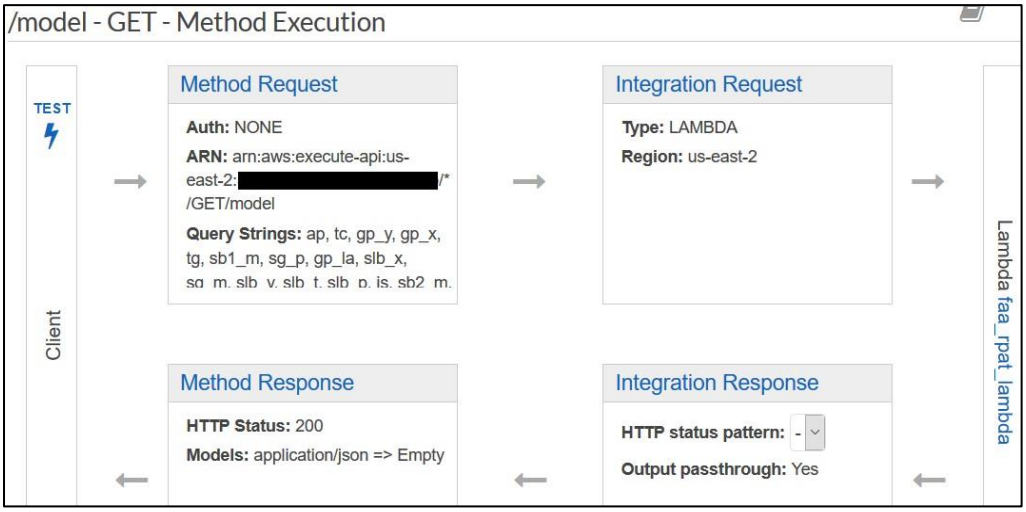


Figure 28: API Gateway - Method Execution

The API takes HTTP GET request whose format is defined in Method Request. Then Integration Request maps the content from GET request to JSON. Upon result from lambda, the Integration Response maps the result to response format mentioned in Method Response and then the HTTP Response is returned to the client.

- **Method Request**

  For the GET Method Request, we use the API Key Required authentication mechanism to validate the source of the GET Request. Also, we validate the string parameters and headers for incorrect inputs or attacks like SQL Injection. We also define the input variables or query string parameters passed along with the

Request. The variables are passed like *www.xyz.com/resource_name?input1=value1&input2=value2*. For RPAT Project we accept the following input variables

Input Variable	Variable Full Form
al	Airlines
ap	Airplane
gp_la	Gear Position - Loading Angle
gp_x	Gear Position - X
gp_y	Gear Position - Y
js	Joint Stiffness
sb1_m	Subbase1 - E Modulus
sb1_p	Subbase1 - Poisson
sb1_t	Subbase1 - Thickness
sb2_m	Subbase2 - E Modulus
sb2_p	Subbase2 - Poisson
sb2_t	Subbase2 - Thickness
sg_m	Subgrade - Modulus
sg_p	Subgrade - Poisson
slb_m	PCC Slab - E Modulus
slb_p	PCC Slab - Poisson
slb_t	PCC Slab - Thickness
slb_x	Slab Size - X
slb_y	Slab Size - Y
tc	Thermal Coefficient
temp	Temperature
tg	Temperature Gradient

Here, the “temp” variable is a Boolean to specify if “tc” and “tg” variables are also provided. Therefore, “tc” and “tg” variables are not required while all other variables are required.

- **Integration Request**

  This module defines how to extract the input variables from the Request and where to forward them for processing. For the RPAT project, it is forwarded to the Lambda function we create in the next steps. Therefore, create the integration request of type lambda in US-East-2 region. Enter the name of lambda function that you will create in step 6.2.2.

  Add the mapping template with the name "*application/json*". This tells AWS to send a JSON containing all variables to the lambda function. Once application/json is available in Content-Type, click on it. This would open a text editor. Enter the following code.

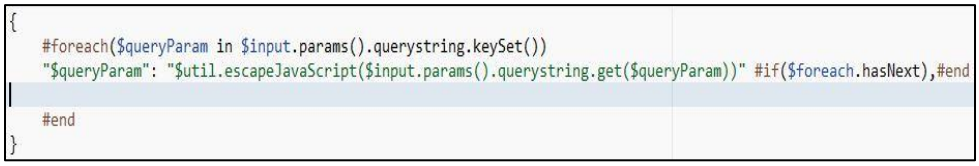


Figure 29: API Gateway - Content Mapping

  This code defines how to map the input variables from the GET request to JSON.

- **Method Response**

  This defines the method's response types, their header, and content types. Add a Response with Status Code as 200. Next, add a header with the name "*Access-Control-Allow-Origin*". Then, add a response model of content type "*application/json*" and model as empty. For every possible status code like 401 (Page not Found), create a new response.

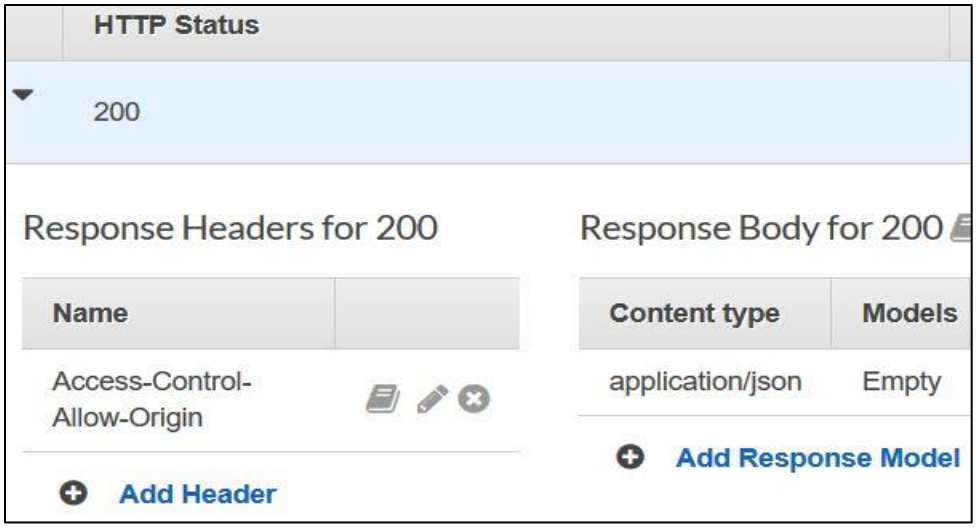


Figure 30: API Gateway - Method Response

- **Integration Response**

  This defines how to map the response data received from a lambda function to the GET response. First, add an integration response with method response status 200 and content handling as Passthrough. For response header, enter mapping value as single quote-star-single quote *'*'*. Add a mapping template with content type as "application/json".

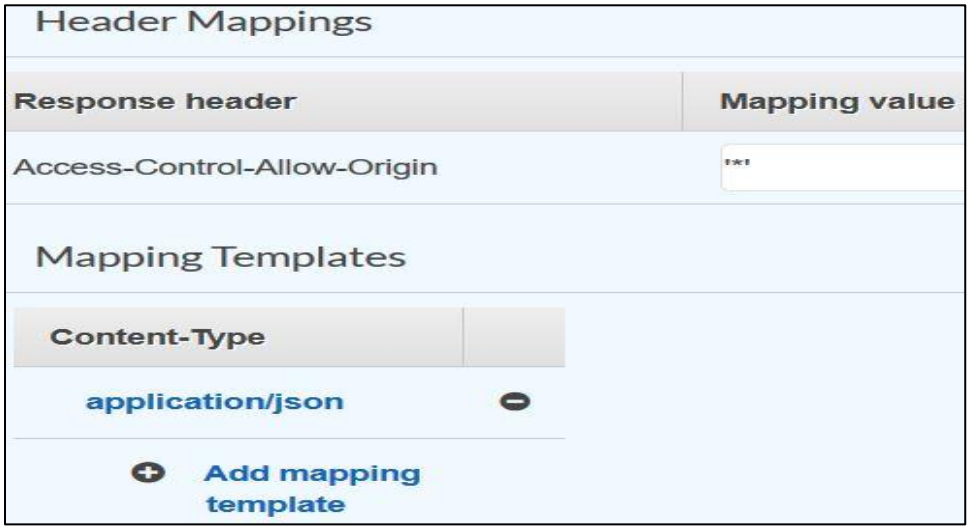


Figure 31: Integration Response

After completing all the 4 steps correctly, you should have a correct API setup.

5.2.2. Lambda Setup

Lambda function is the compute unit for the whole architecture which executes the logic for the application from within the VPC. Since Lambda is partially maintained by AWS, developers can define many settings for lambda execution.

5.2.2.1. Function Creation

AWS provides the lambda service which allows creating lambda functions that execute code from a different programming language. First, create a lambda function using the name we used in the Integrating Response section of API Setup. Define the runtime or programming language for the lambda function. For RPAT, python 3.6 is used. Next, we select to use an existing role which we create in the 6.2.2.2 step. The execution role will be available in the drop-down list once you create it in 6.2.2.2. For RPAT, we use a role named *faa_rpat_lambda_function*. We can complete these steps after role creation.

Once the function is created, we can configure it from the lambda configuration page. Drag the API Gateway from Add Trigger window to center to set the trigger for lambda function. This can also be done from API Gateway by linking the API to lambda in Integration Request menu. In this case, the API Gateway would be automatically added as the trigger.

5.2.2.2. Execution Role

Roles allow the compute services in AWS to act on behalf of a user. Roles have permissions which allow and restrict access to services. We can create a new role in Identity and Access Management (IAM) service of AWS. Create a new role "*faa_rpat_lambda_execute*" for lambda function from the Roles menu in IAM. Next, attach the "*AWSLambdaReadOnlyAccess"* policy and "*AWSLambdaVPCAccessExecutionRole"* policy provided by AWS. The policy contains a set of permissions that are granular to the service operation level.

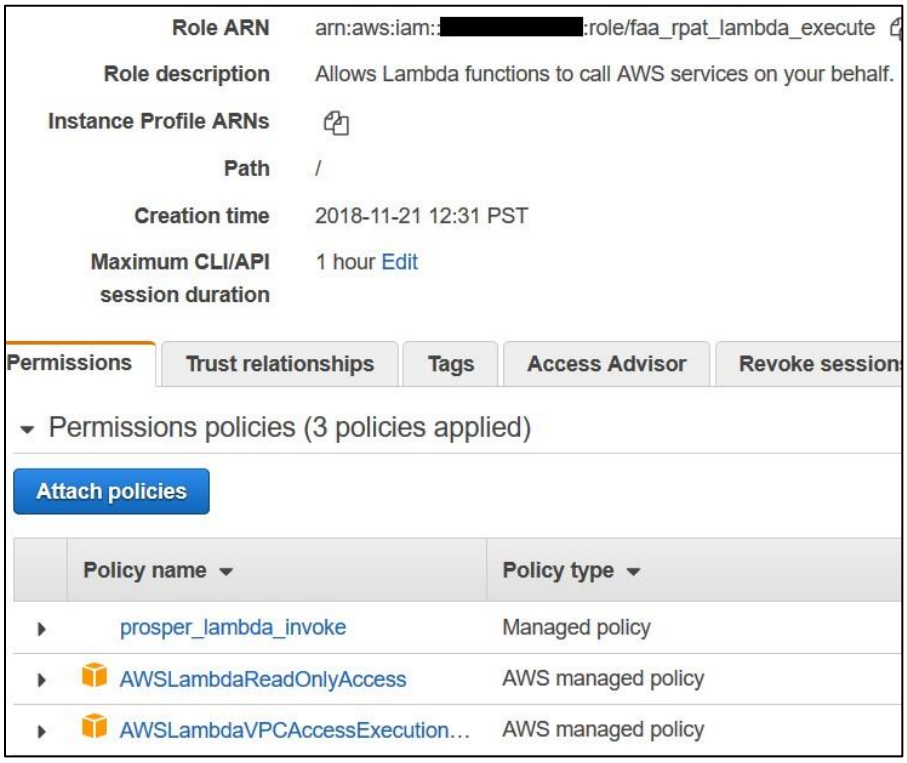


Figure 32: Lambda - Role Details

However, to enable the Lambda to call another lambda function, it requires additional permission. Therefore, we create a custom policy. AWS allows to create a custom

policy using UI or via JSON code. We use the following code to create the *prosper_lambda_invoke* policy and then link it to the *faa_rpat_lambda_execute* role defined above.

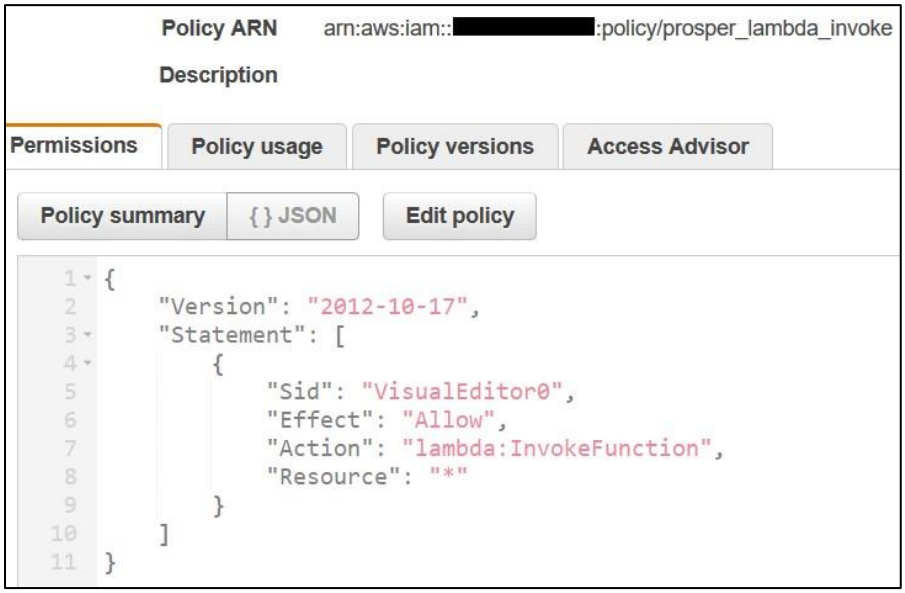


Figure 33: Lambda - Custom Policy

Now you can attach this role to the lambda function from the lambda configuration page of that lambda function.

5.2.2.3. Network Setting

Lambda function could be created within VPC or outside. If done outside, we do not need to define the VPC and the subnet but if done within VPC, then, it requires two subnets and 1 security group.

We create the RPAT lambda function within the VPC using the two subnets we defined before.

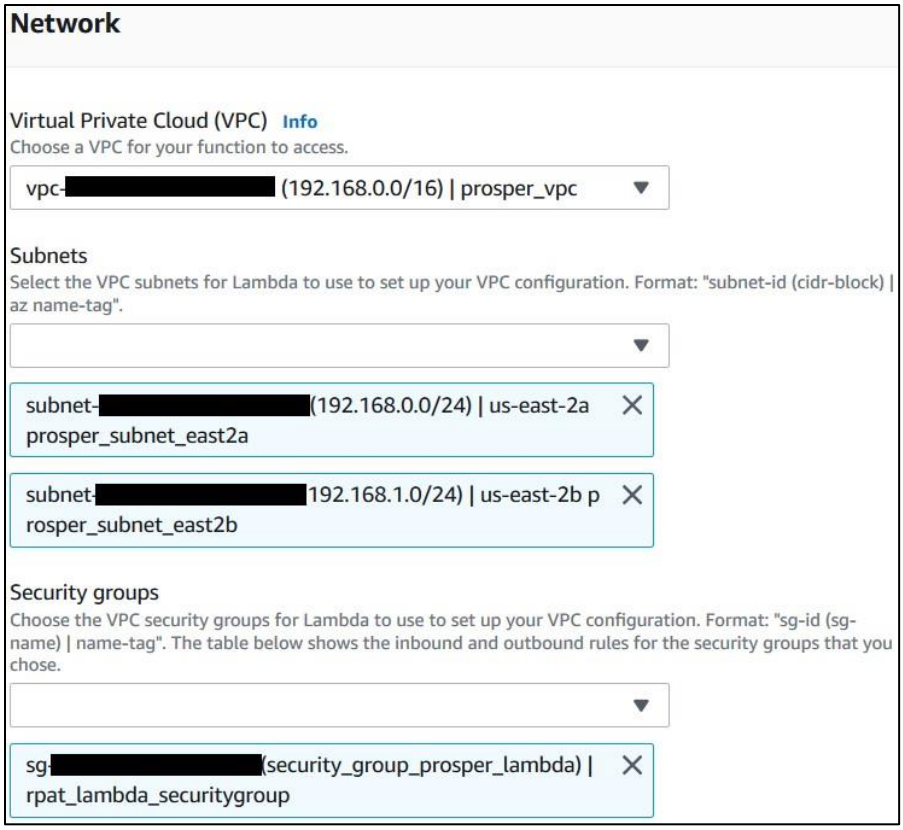


Figure 34: Lambda - Network Details

We define a new security group for the lambda function to control the type of traffic that enters lambda and to make customization for different applications easy.

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	::/0
All TCP	TCP	0 - 65535	192.168.0.0/16
MYSQL/Aurora	TCP	3306	192.168.0.0/16
HTTPS	TCP	443	0.0.0.0/0
HTTPS	TCP	443	::/0

Figure 35: Lambda - Security Group Inbound Rules

The inbound rules are like that of VPC expect the SSH traffic is not allowed. This policy should be updated for different applications based on their needs.

5.2.2.4. Environment Variables

This provides an option to set environment variables which are loaded into the lambda function before it starts the execution.

5.2.2.5. Function Code Upload

AWS provides three ways to upload the code for lambda function. You can upload a zip file, import file from S3 for write code in the inline code editor window. These options vary by the programming language used.

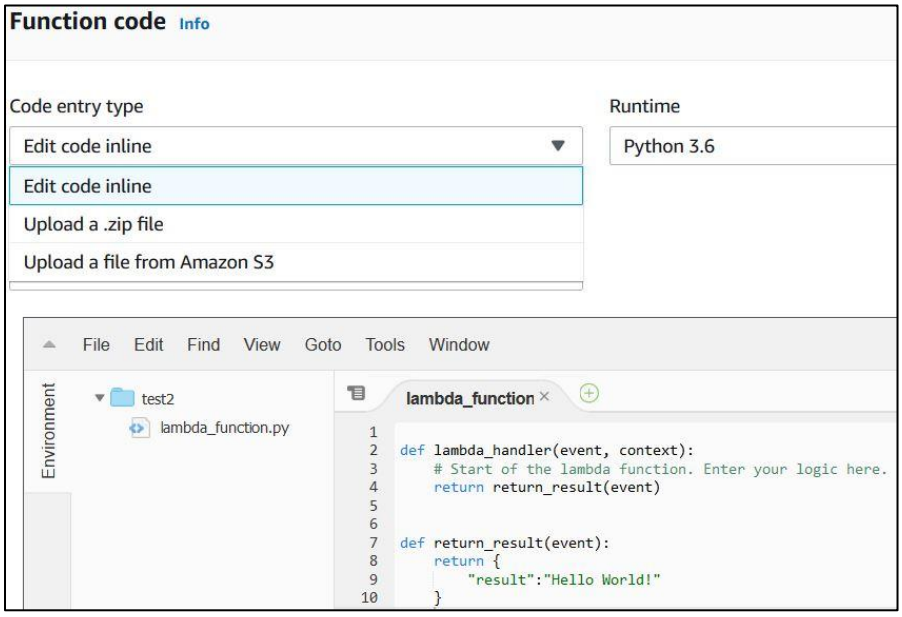


Figure 36: Lambda - Function Code Editor

For C# as the programming language, the visual studio provides an AWS Toolkit to upload the code in a lambda function directly from visual studio available at https://aws.amazon.com/visualstudio/.

For RPAT, we use python where all the logic is defined in the lambda_function.py which is then uploaded to lambda using .zip upload method.

If followed all the steps correctly, the entire AWS architecture will be ready to deploy any application.

5.3. ANDROID APP CREATION

For the RPAT project, we create an Android App as User Interface which allows users to feed in inputs and receive the predicted values. Android App is very convenient for onsite workers as the service is available to them wherever they go. For the development of the App, Android Documentation [13] was referred.

5.3.1. Menu

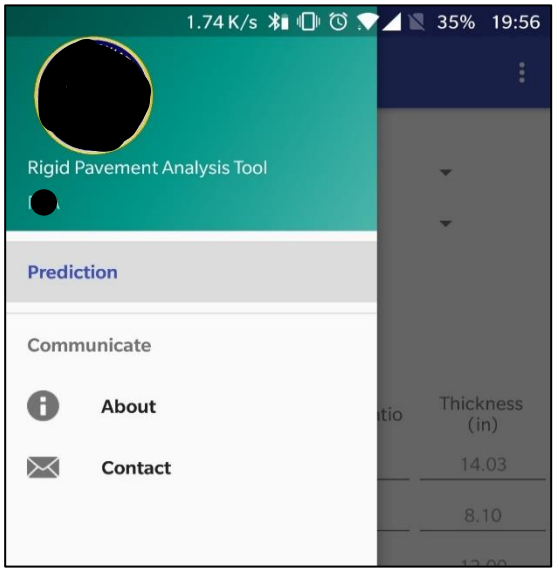


Figure 37: Android - Drawer Menu (Curtesy of ISU PROPSER Team

The Menu provides a quick drawer window to navigate to other resources within the App. It provides a convenient way to visualize what services are available in the App and use them. The RPAT App provides the prediction utility, information about the App and an ability to contact the team for feedback.

5.3.2. About Us Activity

Figure 38: Android - About Us Activity

The About Us page within the App provides the user with a brief introduction to the project.

5.3.3. Prediction Activity

The Prediction Activity is the Main utility of the App. It provides a well-designed user interface to allow users to input the parameters for the prediction model. The inputs are then sent to the cloud specifically the API Gateway via REST for processing. Once the lambda functions complete the processing of the data, the result is sent back to the App with the status of the processing. If the status is a success, then the result is displayed else error message is shown to the user.

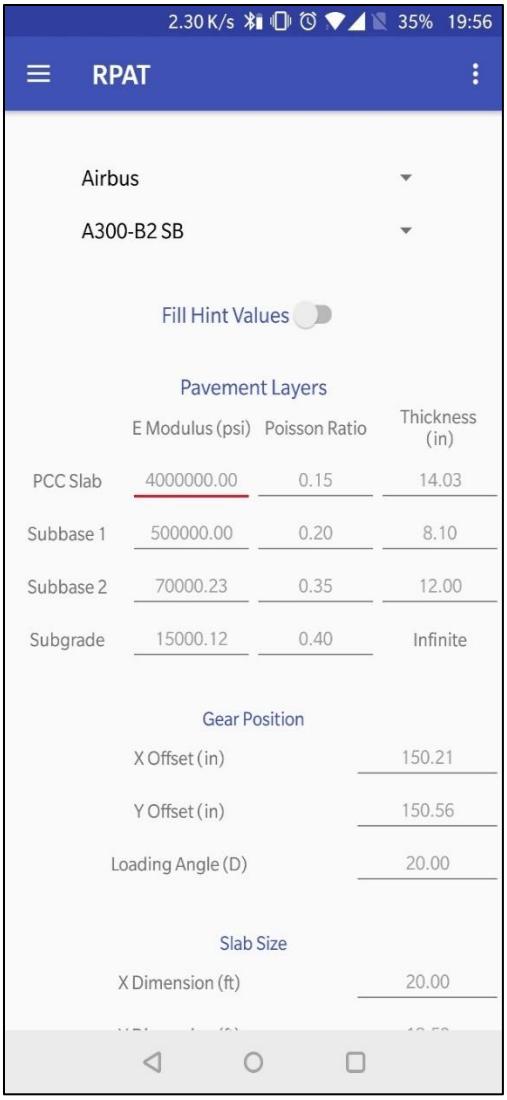


Figure 39: Android – Prediction

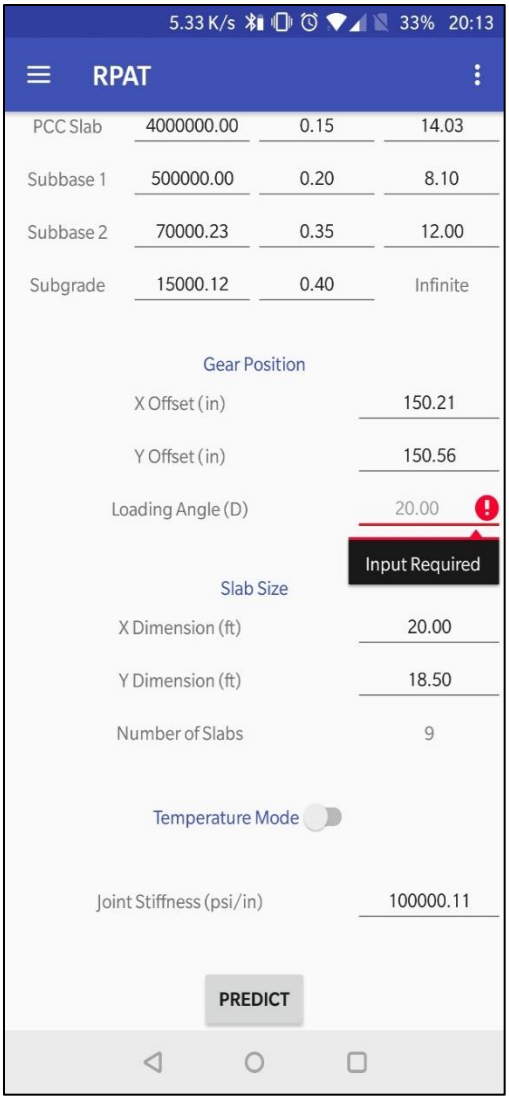


Figure 40: Android – Error Handling

The Prediction activity performs the following tasks

- **Accepts user inputs**

  The UI first provides a dropdown list to select the airplane manufacturing company. Upon selection, the UI populates the dropdown list for the airplane model they want to select. Next, the user must enter the values for 17 features used for predicting the stress. Upon successful insertion of all values, the user presses the Predict button which sends the inputs for

processing to the cloud. Once it receives the result, the Predict activity starts the Result activity which displays the result.

The 17 features are:

Pavement Layer:

1. PCC Slab – E Modulus
2. PCC Slab – Poisson Ratio
3. PCC Slab – Thickness
4. Subbase 1 – E Modulus
5. Subbase 1 – Poisson Ratio
6. Subbase 1 – Thickness
7. Subbase 2 – E Modulus
8. Subbase 2 – Poisson Ratio
9. Subbase 2 – Thickness
10. Subgrade – E Modulus
11. Subgrade – Poisson Ratio

Gear Position

12. X Offset
13. Y Offset
14. Loading Angle

Slab Size

15. X Dimension
16. Y Dimension
17. Joint Stiffness

- **Validates the Inputs**

  Since the UI takes in inputs, they need to be validated for existence, correctness, and a match to the expected value. This helps to prevent attacks like SQL Injection. The UI check all the input fields for all the three conditions. If any of them is false, then it displays an error message as shown above. Only when all the inputs are validated will the Predict button send the data to the server.

- **Toggles the temperature mode**

  The prediction models for RPAT enables users to use 2 more features for better prediction results however they are optional. The Temperature mode toggle button displays them in the UI when the button is toggled on. Inputs for these two inputs are also automatically taken into consideration for validation.

  The two features are

  1. Temperature Gradient
  2. Slab Coefficient

- **Hint expected values**

  The inputs boxes provide a hinted value for the user to understand the expected values. When the user starts entering the data, the hinted value is removed from the display.

- **Autofill hint values for testing.**

  While working onsite or developing a prototype, there often is a need to test the app for connection issues or prediction issues.

The autofill toggle button enables users and developers to quickly test the App without having to enter the 17 feature values every time.

5.3.4. Result Activity

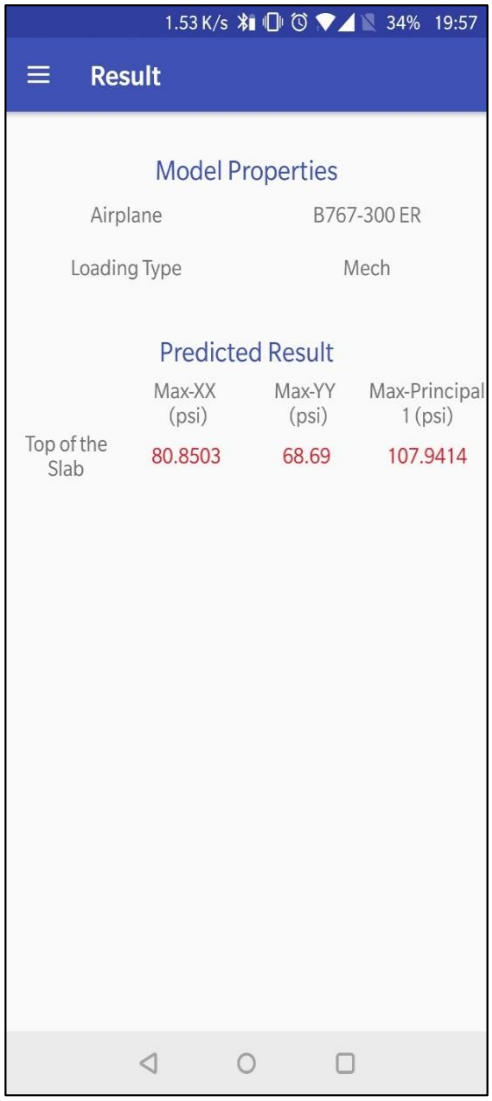


Figure 41: Android - Result Activity

The Result activity is called when the user presses the Predict button in Predict Activity and the cloud return a success status. The activity displays the airplane information and the model type selected by the user and later displays the predicted results.

The result displays the values predicted by the model. Since the task is to predict the force exerted by airplanes on the runway during landing, the three predicted values are namely:

1.Max-XX: Maximum stress along the X-Axis

2.Max-YY: Maximum stress along the Y-Axis

3.Max-Principal: Maximum principal stress

5.3.5. Contact Us Activity

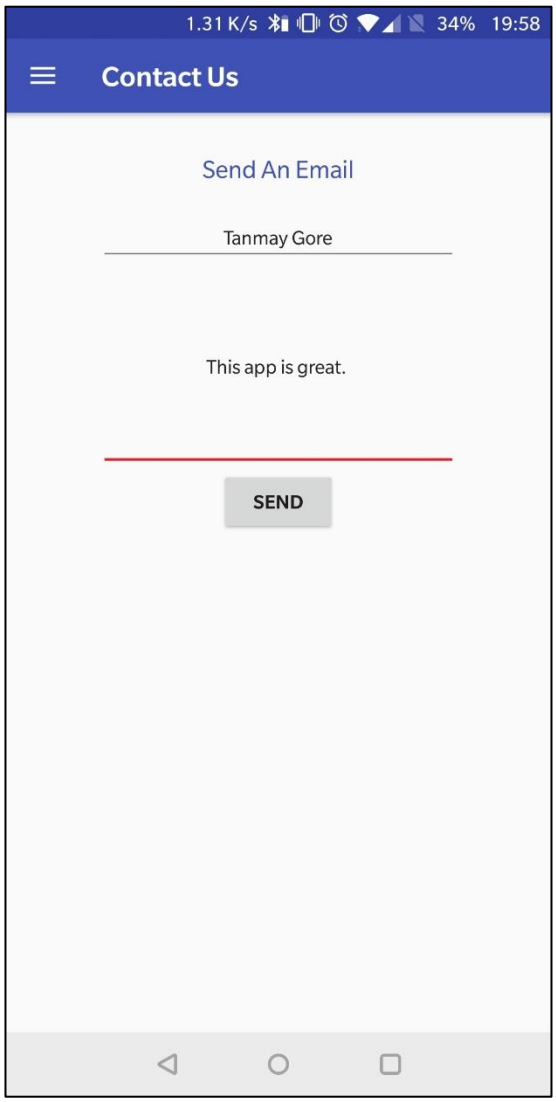


Figure 42: Android - Contact Us Activity

The Contact Us activity allows users to provide feedback about the App. It automatically sends an email using users default email address to the team. The email includes the name and the messages entered by the user in the input boxes.

6. PERFORMANCE ANALYSIS

The new system should perform better than the old system. Therefore, we compare the benefits for the new system with respect to the old one using different statistics and feature comparison

6.1. Comparison to Windows-based Tool

	Windows Based Tool	Cloud Based Tool
Access over Internet	No	Yes
Multi-platform deployment	No	Yes
Model File Security	No	Yes
Access to Tool Failure Logs	No	Yes
Remote Updates	No	Yes
Prediction Time	500 – 800 ms (User System Dependent)	1000 – 1400 ms
Security Over Internet	-	Yes
Monitor Tool Usage	No	Yes
Restrict Tool Usage	No	Yes
Multi-User Support	No	Yes
Monthly Cost to Owner	No	Yes

Table 1: Performance Analysis - Tool Comparison

The cloud-based tool provides a lot of features like platform independence, access over the internet and many more at the cost of few additional milliseconds.

6.2. Lambda Processing Time

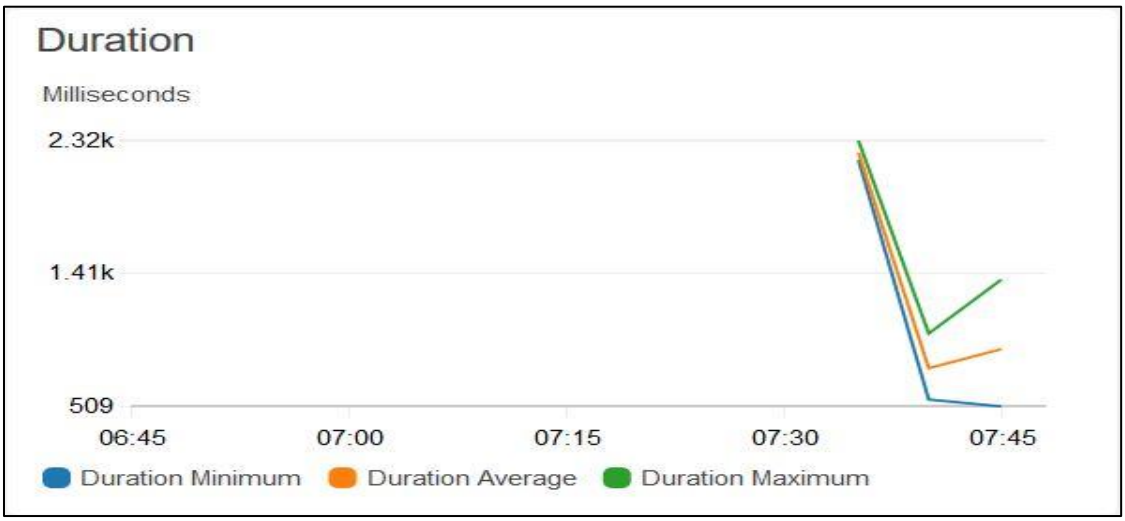


Figure 43: Performance Analysis - Lambda Processing Time

As seen above, the lambda function can scale and automatically reduce the processing time. Lambda function initially takes a few seconds for processing; however, it automatically scales to reduce the processing time for consecutive execution.

6.3. Availability and Error Rate

Figure 44: Performance Analysis - Error and Availability Rate

As seen above, the error rate is zero and the availability is very high. This is because AWS promises about 99.9% availability by automatically handling failures.

6.4. Service Usage

Top Free Tier Services by Usage — View all

Service	Free Tier usage limit	Month-to-date usage
Amazon Relational Database Service	750 hours of Amazon RDS Single-AZ db.t2.micro Instances	24.40% (183.00/750 Hrs)
Amazon Elastic Compute Cloud	750 hours of Amazon EC2 Linux t2.micro instance usage	23.20% (174.00/750 Hrs)
Amazon Elastic Compute Cloud	30 GB of Amazon Elastic Block Storage in any combination of General Purpose (SSD) or Magnetic	6.24% (1.87/30 GB-Mo)
Amazon Simple Storage Service	20,000 Get Requests of Amazon S3	0.87% (174.00/20,000 Requests)
AmazonCloudWatch	5 GB of Log Data Ingestion for Amazon Cloudwatch	0.57% (0.03/5 GB)

Figure 45: Performance Analysis - Service Usage as of March 8th

Since the AWS architecture is currently set up on an AWS Free Tier eligible account, few services have a free usage limit. The application currently works entirely within the free tier.

6.5. Cost of Deployment

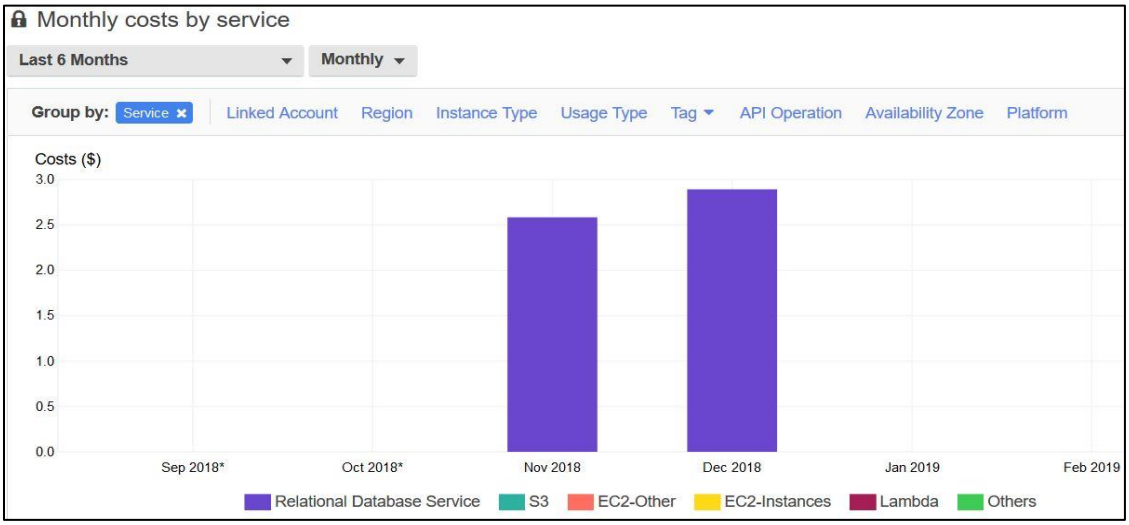


Figure 46: Performance Analysis - Cost by Service

Since September 2018, the cost for the cloud services has been almost zero. Since the RDS replication across multiple regions was turned on for November and December, there were charges of amount $3 which is minimal for the benefits of the system.

7. FUTURE WORK

There are many ways to improve this system even better.

Following are the improvements within the Cloud.

1. Reducing the processing time of lambda functions when called for the first time after a while.
2. Making the lambda deployment process even simpler by defining roles and policies that will work for any application.
3. Including other services provided by AWS within the VPC to help deploy the application without worrying about adding the services themselves.
4. Create terraform code to quickly recreate the platform on other AWS accounts
5. Create logic to automatically update the model files in the S3 and RDS.

Following are the improvements within the Android App.

1. Adding new features like storing the results for later viewing and creating graphs to visualize the changes in result values.
2. Making Android App more secure and private by adding User Authentication within it.
3. Making the app more accessible to disabled people using voice to text and text to voice to enter the parameter values.

8. CONCLUSION

The AWS Cloud-based application deployment platform was able to meet all the design goals to ensure that developers can develop their logic for machine learning based application without worrying about deployment-related issues.

The use of REST framework along with API Gateway allows the server to communicate with any UI device in a standard way. Thus, providing cross-platform support. Also, API gateway provides GUI for defining APIs and flow of program making API development very easy. Lambda functions are partially managed by AWS, can run programs written in many languages and automatically scale to provide better performance. Thus, providing Multi-language support, Scalability and High Availability of the application. The VPC takes care of security and privacy for the whole system. The security groups, access control lists, permissions, and roles enable applications to utilize AWS provided services in a secure fashion. The platform provides an option for developers to fully customize the architecture to suit their needs. Therefore, it provides Privacy, Total Control, Security, Flexibility, and Customizability.

Finally, we successfully migrate the Rigid Pavement Analysis Tool from windows to the platform, develop an android app with an extensive focus on user-friendly design and provide good performance to ensure its utility in real-world scenarios.

9. REFERENCES

[1] "What is Cloud Computing?," Microsoft, [Online]. Available: https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/. [Accessed 14 March 2019].

[2] "Cloud Computing with Amazon Web Services," Amazon, [Online]. Available: https://aws.amazon.com/what-is-aws/. [Accessed 14 March 2019].

[3] "What is AWS Lambda," Amazon, [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. [Accessed 14 March 2019].

[4] "What is Amazon API Gateway," Amazon, [Online]. Available: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html. [Accessed 14 March 2019].

[5] "What is Amazon VPC," Amazon, [Online]. Available: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html. [Accessed 14 March 2019].

[6] "What is Amazon Relational Database Service (Amazon RDS)," Amazon, [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html. [Accessed 14 March 2019].

[7] "What is Amazon S3," Amazon, [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html. [Accessed 14 March 2019].

[8] "Portland Cement Concrete," Farlax, Inc, [Online]. Available: http://encyclopedia2.thefreedictionary.com/Portland+Cement+Concrete. [Accessed 14 March 2019].

[9] "Subbase (pavement)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Subbase_(pavement). [Accessed 14 March 2019].

[10] "Subgrade," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Subgrade. [Accessed 14 March 2019].

[11] "AWS Documentation," Amazon, [Online]. Available: https://docs.aws.amazon.com/index.html#lang/en_us. [Accessed 14 March 2019].

[12] "AWS Management Console," Amazon, [Online]. Available: https://aws.amazon.com/console/. [Accessed 2019 March 2019].

[13] "Documentation for App Developers," Google, [Online]. Available: https://developer.android.com/docs/. [Accessed 14 March 2019].

[14] "MySQL Documentation," Oracle, [Online]. Available: https://dev.mysql.com/doc/. [Accessed 14 March 2019].

10. ACKNOWLEDGMENT

I would like to express my deepest appreciation to my major advisors, Dr. Halil Ceylan, and Dr. Wensheng Zhang for their guidance, encouragement, and support throughout this project. Without their guidance and support, this creative component project would not have been possible. My sincere thanks also go to Dr. Sunghwan Kim for his invaluable guidance and assistance throughout the project. In addition to my major advisors, I would also like to thank my committee member, Dr. Shashi Gadia for their guidance during my project selection and their efforts towards my final oral exam. I also want to thank PROSPER and INTRANS for providing me the opportunity to use their work and resources as part of my project. I want to thank my lab mates and colleagues Mr. Adel Rezaei-Tarahomi, Mr. Orhan Kaya and Mr. Yogiraj Sargam for their continuous assistance throughout the project. I would also like to thank my family for their deepest love, support, and trust which helped me get through my years of study.

11. APPENDIX A: ADD/UPDATE MODEL FILES

For the RPAT Project, the data is stored in two places. The Neural Network Models are stored in S3 bucket due to their huge size. While the metadata about planes and their respective models is stored in the RDS MySQL database. The MySQL database contains two tables, one to store model information and the other to store the airplane information along with links to model for those airplanes. Here we will see how to add new model files into the two databases for any airplane. Unfortunately, this process currently is manual and could be made automatic in future work on the project.

11.1. Add/Update Files to S3

1. Upload files to S3 bucket using GUI

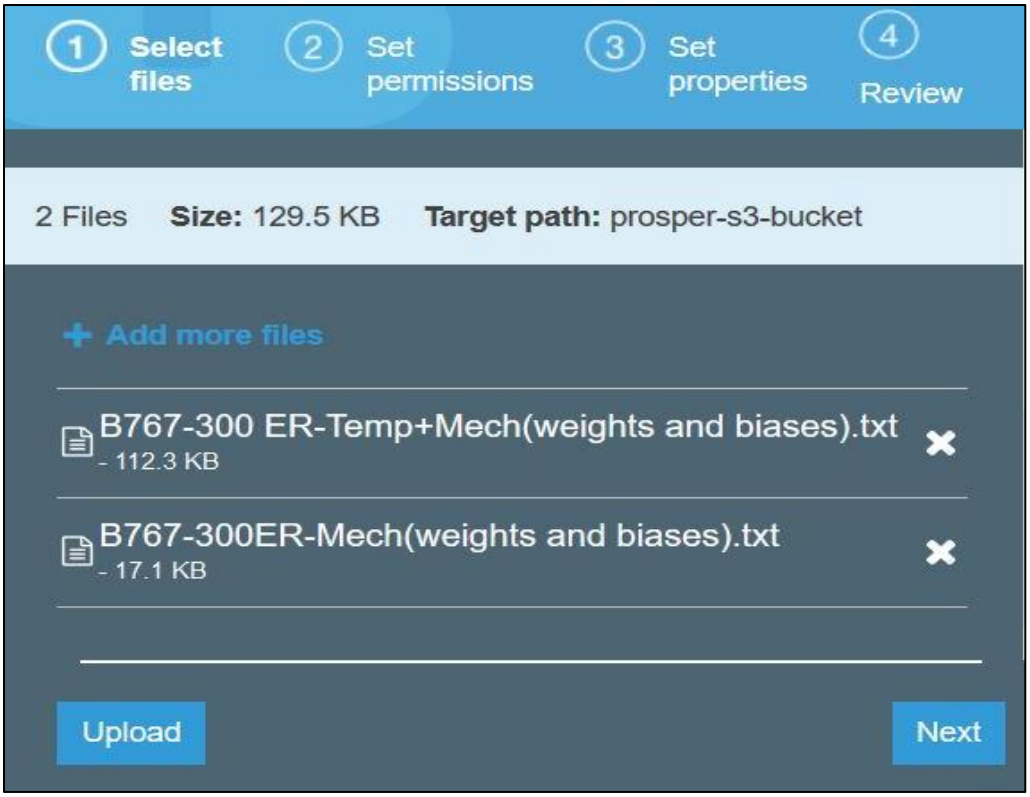


Figure 47: S3 - Upload Files - Select Files

From S3 bucket, select the model files you need to upload. If you are updating the model files, ensure the name of the old and new file is the same. S3 automatically updates the file with the same name using its version number concept.

2. **Set permissions to uploaded files**

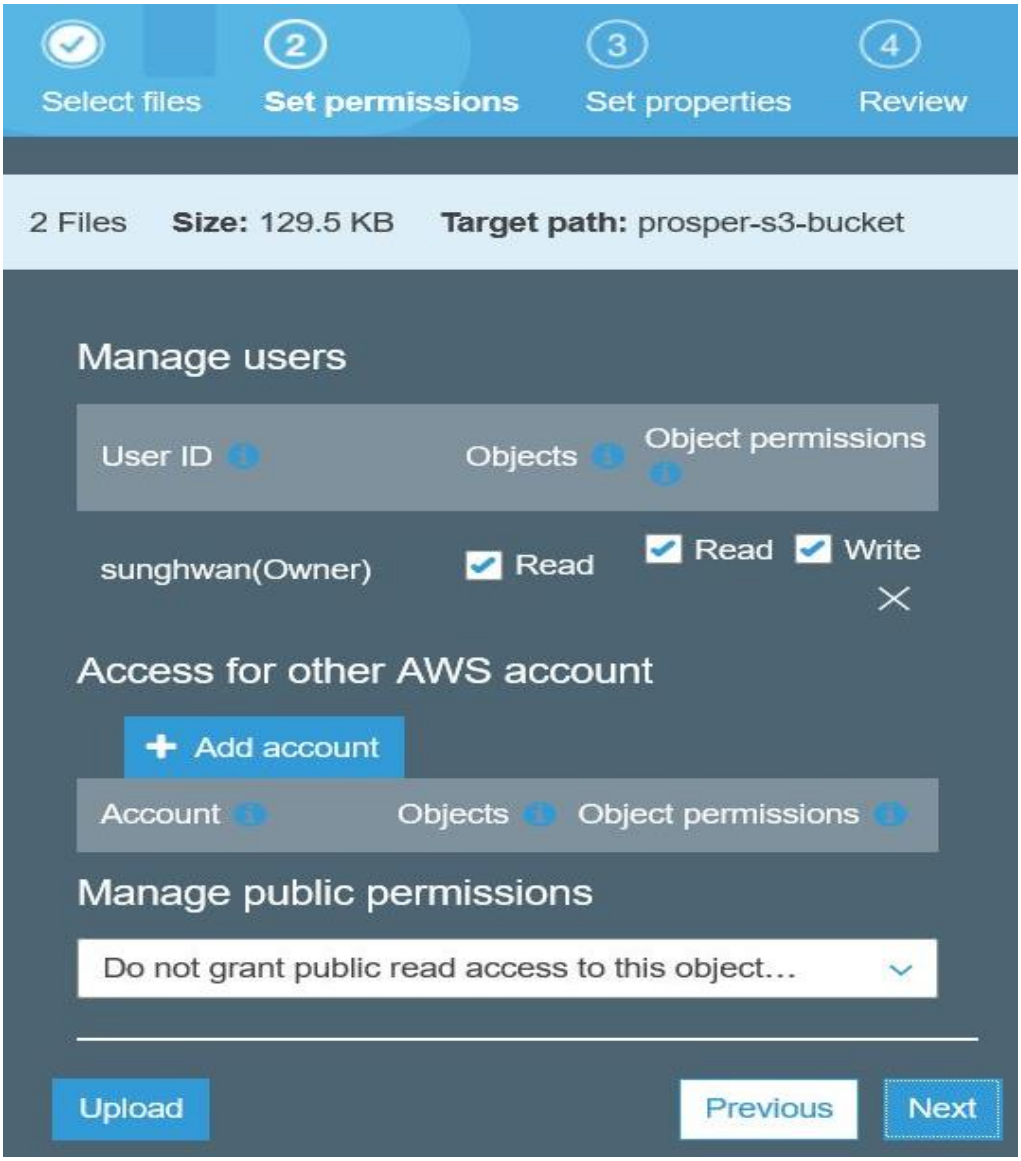


Figure 48: S3 - Upload Files - Set Permissions

We need to set appropriate permissions to enable the user to access the files. This also allows you to share the files with other AWS

accounts. For security, we are not granting public access to files and only allowing uploader to access the files.

3. **Set properties of uploaded files**

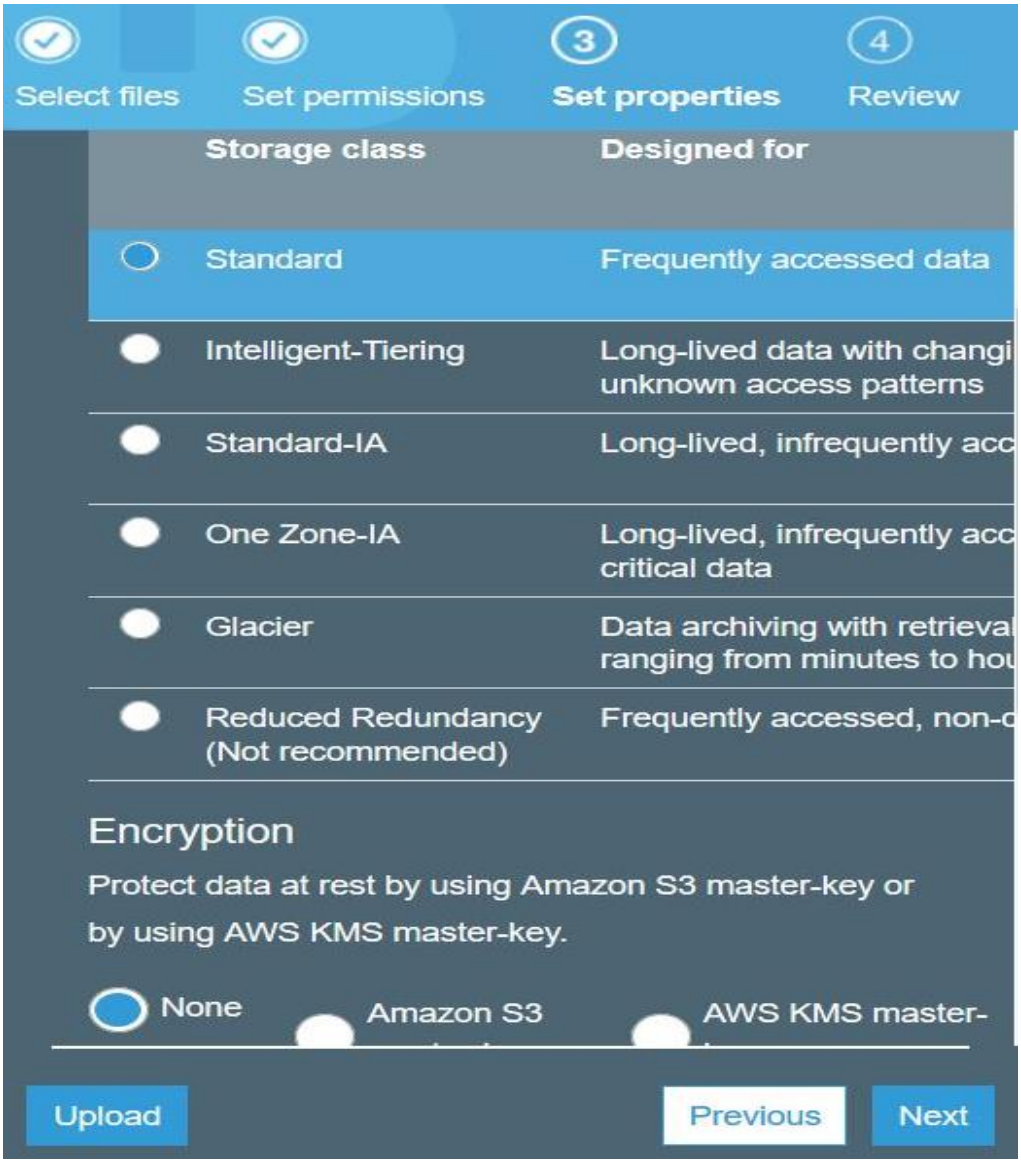


Figure 49: S3 - Upload Files - Set Properties

Next, we set the file as frequently accessed with None encryption. By default, S3 files are encrypted with AES 256-bit encryption. Therefore, none value does not overwrite that encryption setting.

4. **Review Properties**

   Lastly, review the properties and upload the files. This should upload the files to S3.

11.2. Add/Update records in RDS

1. **Record version number and name of files uploaded in S3**

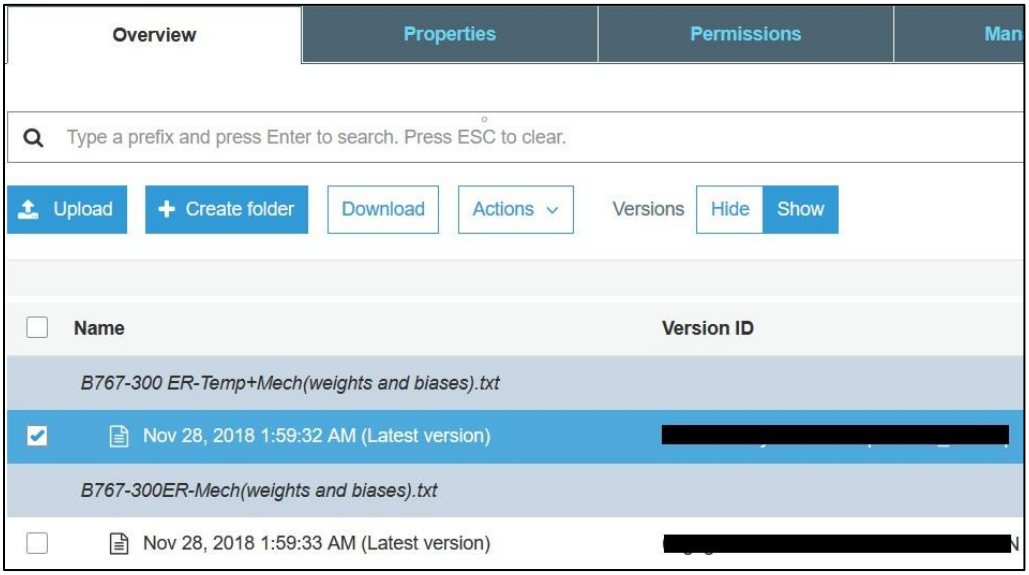


Figure 50: S3 - Upload Files - Version Number

For the files uploaded in the S3 bucket, record the version ID and the file name as it would be required to update or insert records into RDS tables.

2. **Connect to EC2 instance via PuTTY**

   Since RDS is hidden behind the VPC, you cannot directly update the records. Since you can only access RDS from within VPC, we first connect to the EC2 instance within VPC using SSH protocol. We use PuTTY tool which helps to connect using SSH.

First, you need to provide the location of the private key AWS provided while creating the EC2 instance. This key is of .pem format which needs to converted to .ppk format as mentioned here https://tecadmin.net/convert-pem-to-ppk-private-key/. Next, provide this key location in PuTTY software.

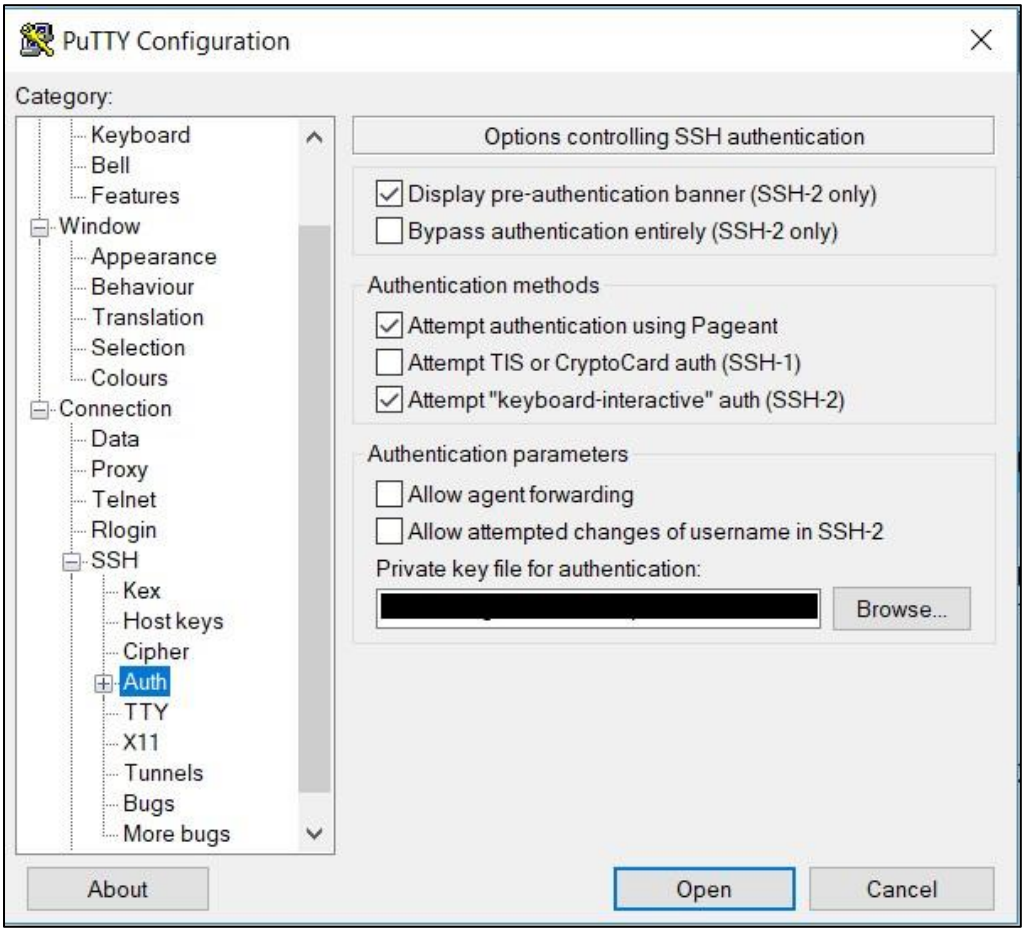


Figure 51: EC2 - PuTTY Authentication Setup

Provide the location of .ppk file at Connection -> SSH -> Auth -> Browse. Next, you need to connect to EC2 instance. In Sessions, provide the Host Name or Public IP Address of the EC2 instance. This information will be available in the EC2 service information on the AWS Management Console.

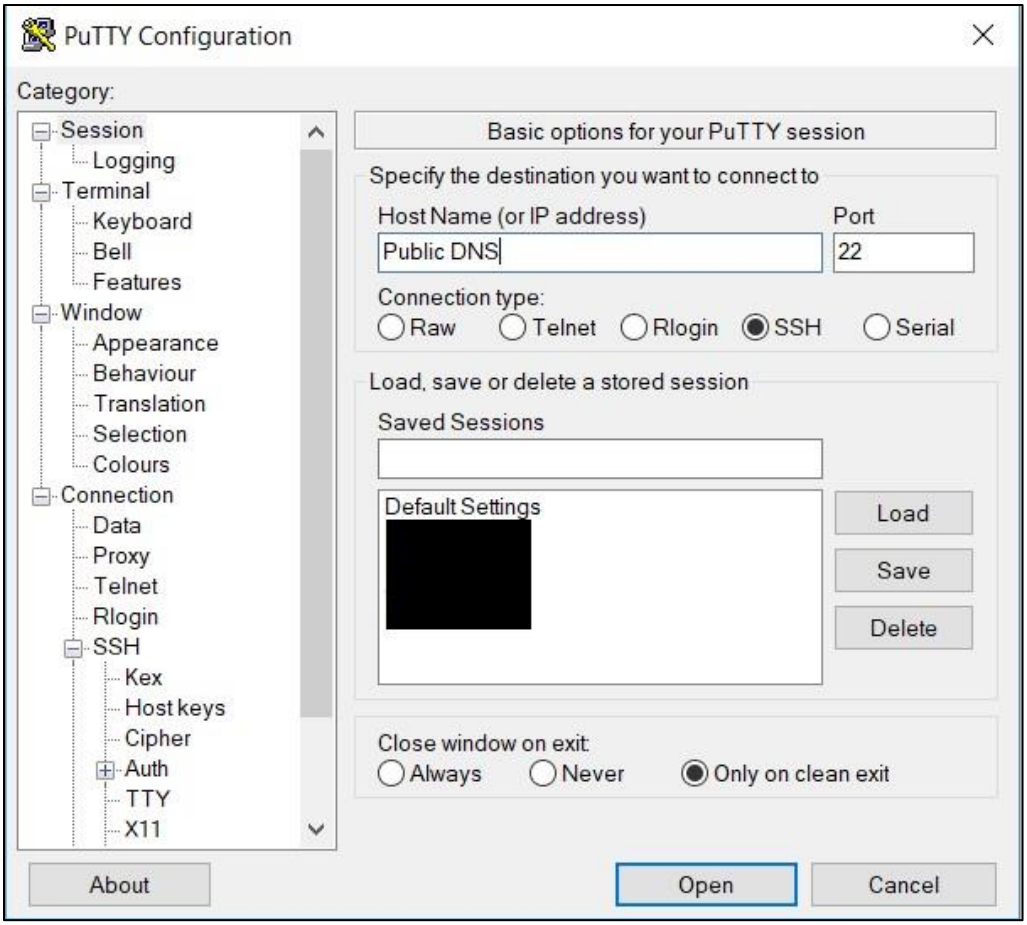


Figure 52: EC2 - PuTTY Connection Setup

For added security, the security group for VPC, only allow the IP Address of specific computer to access the EC2 instance. Therefore, you will need to update the security group rule where the incoming IP Address for SSH should be your public IP Address. You can view your public IP Address by searching “My Ip address” on Google.

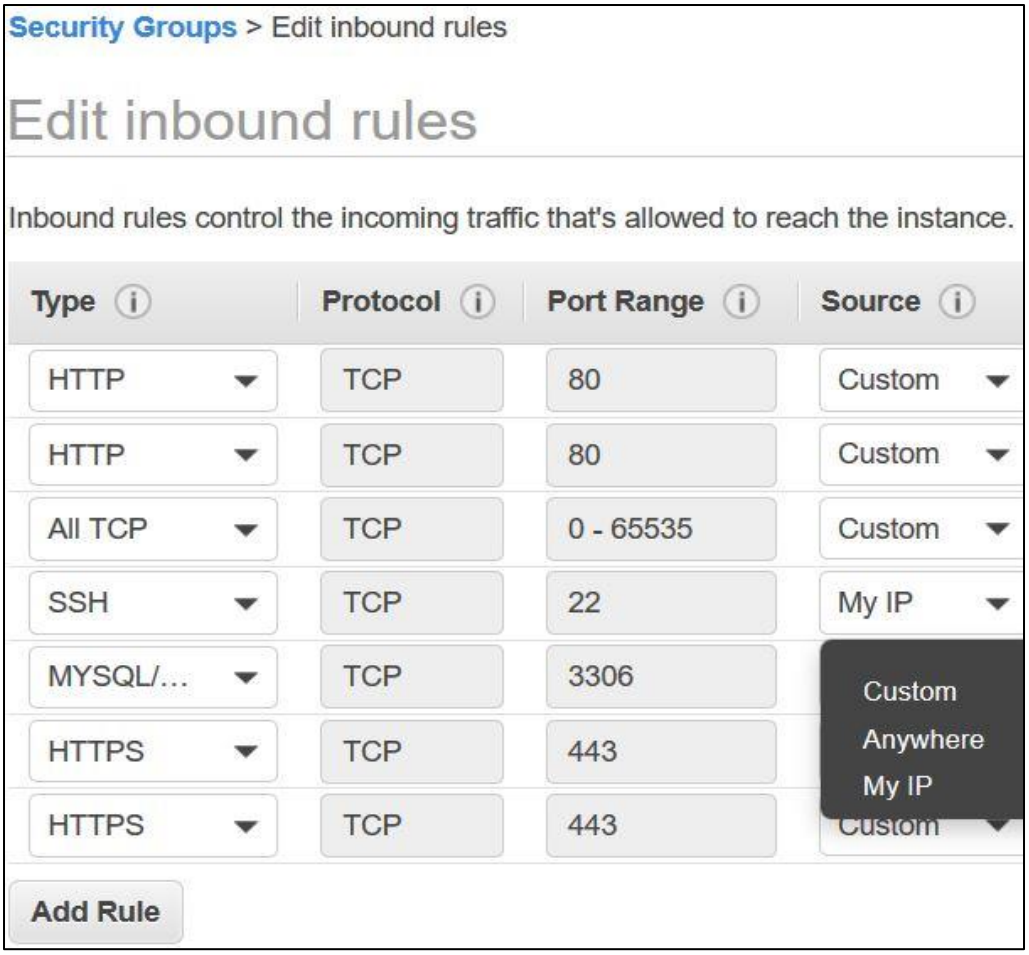


Figure 53: Security Group - VPC - SSH Rule

Once done correctly, a command line window will appear asking for a username. Insert "ec2-user" as the user name.

3. Connect to RDS MySQL database from within EC2 instance

Once you are connected to EC2 and the command prompt open, you can control the EC2 instance using the command prompt.

Next, you need to connect to the RDS MySQL Database [14]. For that enter the following command:

```
mysql -h prosper-faa.xxxxxxxxxxxx.us-east-
2.rds.amazonaws.com -P 3306 -u prosper2018 -p
```

The -h parameter takes the hostname of the service. In this case, it is the Endpoint of the MySQL database. The -P takes the port number. By default, MySQL uses 3306 port. The -u takes the username for the database. -p takes the password for the database. All this information is available on the AWS Management Console under the RDS service page.

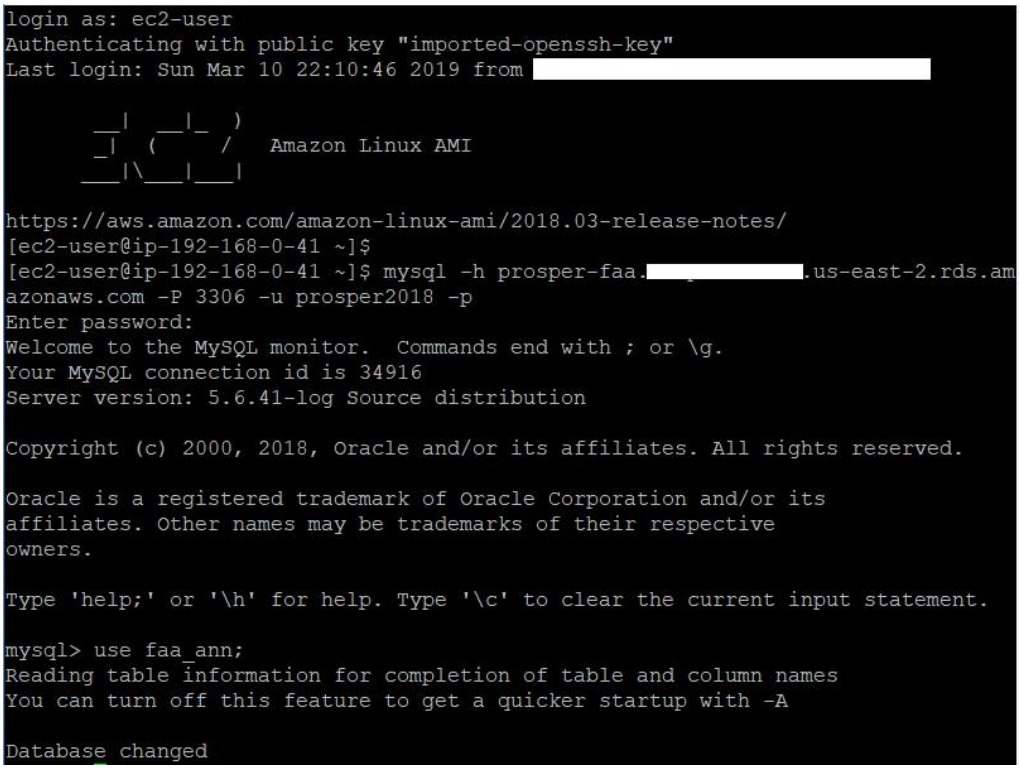


Figure 54: RDS - Connect from EC2

Since we did not provide the password after -p parameter, command prompt asked to enter the password.

Next, you need to select the database. Therefore, we enter the command “use faa_ann;”. Faa_ann is the name of the database. Now we can view, insert and update tables in the database.

4. **Insert/Update record in Models Table**

   Since the plane-info table is dependent on the model table for the model ID, we insert/update the model table first.

   If we are inserting a new model, then use the following SQL Code:

```
insert into models (model_file, version_id, model_type)
values ('model file name', 'version ID of file', 'mech or
temp+mech');
```

   If we are updating the existing model, then we just need to update the version_id using following SQL Code:

```
update models
set version_id = "new version id"
where model_file = "model file name without format";
```

   The model file name does not contain the file type extension like ".txt". Model type is either "mech" or "temp+mech" based on what is mentioned in the file name. Version id should be matching the value from S3 bucket. Now we have inserted or updated the model file. Next, we need to reflect this change in the plane_info table.

5. **Insert/Update record in Plane_Info Table**

   If we updated the model file before, we do not need to change anything here. Since plane_info table links the model files using the model_id, the model_id did not change just the version_id changed.

   If we inserted a new model file for a new airplane, then we need to add the information for the new airplane and link the model file to it. Therefore, we use the following code.

```
insert into plane_info (company, airplane, mech_model_id,
mechtemp_model_id)
values ('Boeing', 'B767-300 ER', 1, 2);
```

Here, for the company, insert the correct company name. For the airplane, insert the correct airplane name. For every airplane, there are two models of type “mech” and “temp+mech”. Therefore, for mech_model_id and mechtemp_model_id, insert the correct model id from the Models table. You will not be able to update the plane_info table until both mech and mech+temp model for a given plane exists in the Models table. The table requires both the ids to successfully update or create a record.

12. APPENDIX B: CREATING NEW APPLICATION

Creating a new application on this platform is simple. Assuming you already have the cloud setup correctly, you need to follow the following steps:

12.1. Create development Plan

First, you need to define the components of the application. RPAT project had the UI, the server program, the database for mapping information and a file storage system. Next, identify how to define the components using AWS services. For RPAT, lambda function was used for server program, RDS was used for database and S3 was used for file storage.

Once the components are defined, create a development schedule based on the available time and prioritize features of the application based on importance.

12.2. Define the API

Now for the communication messages, the UI will send to the server, define an API resource and method. For each communication, the input variables or query parameters are going to change but the rest of the process mentioned in 5.2.1 is going to remain the same.

12.3. Create a Lambda Function and link it to API

Now for each API resource and method, we need a lambda function to process the input. One lambda function can handle multiple APIs. Follow the steps from 5.2.2.1 to 5.2.2.4. This will create a lambda function with the VPC and be able to receive data sent via HTTP GET to the API resource.

12.4. Add new services to VPC

Next, if the application is going to require any service not defined above, then you will have to add those services within VPC. If the services are managed by AWS and therefore could not be placed into VPC, then you need to create the service and create a VPC-Endpoint to enable secure communication from within VPC.

12.5. Create and Add data into Database

Next, create the database and the tables within the database. Once the tables are created, you need to populate it with all the data. To add the data into these databases, you will need to use the EC2 instance as mentioned in Appendix A.

12.6. Write the lambda function code

Now that all the services are set up and all the data is available for processing, you need to create the code in any programming language of choice that is supported by lambda function.

12.7. Upload the code to the lambda function

Lambda provides various ways to upload the code logic as mentioned in 5.2.2.5. Once the logic is defined, upload the code.

12.8. Test using API Gateway

API gateway provides a testing tool for each API method where it takes test values for the input variables and displays the returned output. Use this tool or any other REST Testing tool to send HTTP requests and check if the lambda function is returning correct output.

12.9. Deploy the API

Once the API is correctly defined and tested, then it could be deployed to UI devices to use. API Gateway provides actions to deploy the API.

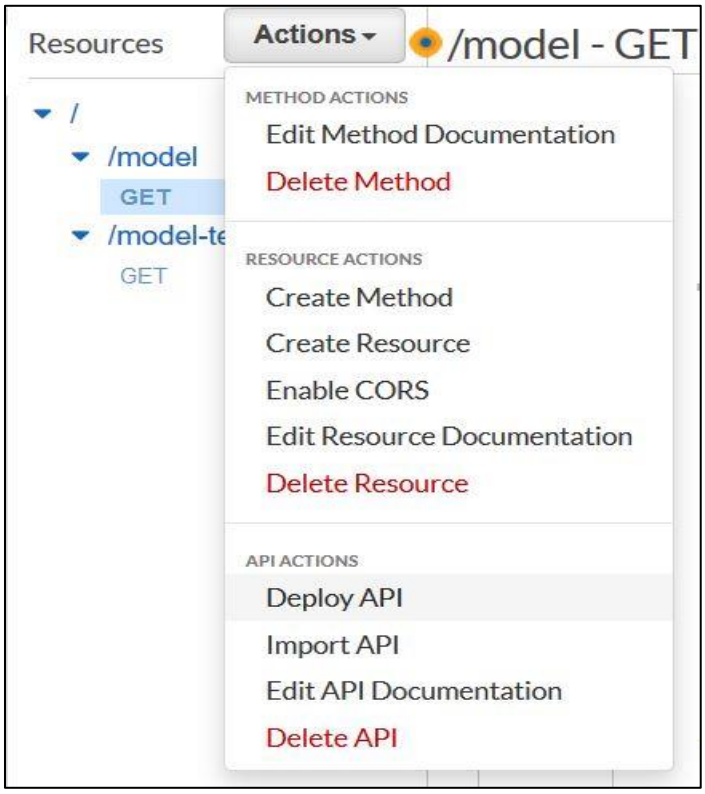


Figure 55: API Gateway - Deploy API

You can deploy to different stages. The Dev stage is for development and used when the development of the application is active. The Prod stage is for production and used when the development is complete, and the application is ready for the people to use.

12.10. Create UI on the preferred platform

Next, you to create the UI for the application on any platform of choice. This could be a mobile app, website or sensor for IoT. UI allows users to interact with the system.

12.11. Link UI to Cloud

The last step is to link the UI with the cloud. Since we use REST, the UI device needs to send the HTTP request whose structure match that of the API method defined before. The UI sends this message only when it needs to send data for processing. Once the request is sent and correctly processed, the server or cloud sends a response. The UI device then must extract data from this response and display or use it as per need. Once this last communication part is successfully created and tested, the application is ready for use. If new changes or new features need to be added into this application, then start the whole process from the beginning. The only difference would be to use the existing resources instead of creating a new one.